



# Arm<sup>®</sup> Cortex<sup>®</sup>-A65AE Core

Revision: r1p1

## Technical Reference Manual

### Non-Confidential

Copyright © 2018, 2020–2022 Arm Limited (or its affiliates).  
All rights reserved.

### Issue 02

101385\_0101\_02\_en



## Arm® Cortex®-A65AE Core Technical Reference Manual

Copyright © 2018, 2020–2022 Arm Limited (or its affiliates). All rights reserved.

### Release Information

#### Document history

Issue	Date	Confidentiality	Change
0000-00	6 December 2018	Confidential	First release for r0p0
0100-00	26 February 2020	Confidential	First release for r1p0
0100-01	12 February 2021	Non-Confidential	First Non-Confidential release for r1p0
0101-02	31 March 2022	Non-Confidential	First Non-Confidential release for r1p1

### 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, has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND

REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

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

Copyright © 2018, 2020–2022 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

## Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

## Product Status

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

## Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>.

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

## Inclusive language commitment

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

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

To report offensive language in this document, email [terms@arm.com](mailto:terms@arm.com).

# Contents

<b>1 Introduction.....</b>	<b>15</b>
1.1 Product revision status.....	15
1.2 Intended audience.....	15
1.3 Conventions.....	15
1.4 Additional reading.....	17
<b>2 Functional description.....</b>	<b>19</b>
2.1 Introduction.....	19
2.1.1 About the core.....	19
2.1.2 Features.....	21
2.1.3 Split-Lock.....	22
2.1.4 Implementation options.....	23
2.1.5 Supported standards and specifications.....	24
2.1.6 Test features.....	25
2.1.7 Design tasks.....	25
2.1.8 Product revisions.....	26
2.2 Technical overview.....	26
2.2.1 Components.....	26
2.2.2 Interfaces.....	30
2.2.3 About system control.....	30
2.2.4 About the Generic Timer.....	31
2.3 Clocks, resets, and input synchronization.....	31
2.3.1 About clocks, resets, and input synchronization.....	31
2.3.2 Asynchronous interface.....	31
2.4 Power management.....	32
2.4.1 About power management.....	32
2.4.2 Voltage domains.....	32
2.4.3 Power domains.....	33
2.4.4 Architectural clock gating modes.....	36
2.4.5 Power control.....	38
2.4.6 Core power modes.....	38
2.4.7 Thread power modes.....	44

2.4.8 Relationship between power modes and power domains.....	45
2.4.9 Power down sequence.....	46
2.4.10 Debug over powerdown.....	47
2.5 Memory Management Unit.....	47
2.5.1 About the MMU.....	47
2.5.2 TLB organization.....	48
2.5.3 TLB match process.....	50
2.5.4 Translation table walks.....	51
2.5.5 MMU memory accesses.....	52
2.5.6 Responses.....	53
2.6 L1 memory system.....	56
2.6.1 About the L1 memory system.....	56
2.6.2 Cache behavior.....	57
2.6.3 L1 instruction memory system.....	60
2.6.4 L1 data memory system.....	61
2.6.5 Data prefetching.....	64
2.6.6 Direct access to internal memory.....	65
2.7 L2 memory system.....	72
2.7.1 About the L2 memory system.....	72
2.7.2 Optional integrated L2 cache.....	72
2.7.3 Support for memory types.....	73
2.8 Reliability, Availability, and Serviceability (RAS).....	73
2.8.1 Cache ECC and parity.....	74
2.8.2 Cache protection behavior.....	74
2.8.3 Uncorrected errors and data poisoning.....	76
2.8.4 RAS error types.....	77
2.8.5 Error synchronization barrier.....	78
2.8.6 Error recording.....	79
2.8.7 Error injection.....	80
2.9 Generic Interrupt Controller CPU interface.....	82
2.9.1 About the Generic Interrupt Controller CPU interface.....	82
2.9.2 Bypassing the CPU interface.....	83
2.10 Advanced SIMD and floating-point support.....	83
2.10.1 About the Advanced SIMD and floating-point support.....	83
2.10.2 Accessing the feature identification registers.....	84
2.11 Split-Lock feature.....	84

2.11.1 Implementing Split-Lock.....	84
-------------------------------------	----

### **3 Register descriptions..... 91**

3.1 AArch64 system registers.....	91
3.1.1 AArch64 registers.....	91
3.1.2 AArch64 architectural system register summary.....	92
3.1.3 AArch64 IMPLEMENTATION DEFINED register summary.....	96
3.1.4 AArch64 registers by functional group.....	97
3.1.5 ACTLR_EL1, Auxiliary Control Register, EL1.....	101
3.1.6 ACTLR_EL2, Auxiliary Control Register, EL2.....	102
3.1.7 ACTLR_EL3, Auxiliary Control Register, EL3.....	104
3.1.8 AFSR0_EL1, Auxiliary Fault Status Register 0, EL1.....	106
3.1.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2.....	106
3.1.10 AFSR0_EL3, Auxiliary Fault Status Register 0, EL3.....	107
3.1.11 AFSR1_EL1, Auxiliary Fault Status Register 1, EL1.....	107
3.1.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2.....	107
3.1.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3.....	107
3.1.14 AIDR_EL1, Auxiliary ID Register, EL1.....	108
3.1.15 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register, EL1.....	108
3.1.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2.....	108
3.1.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3.....	108
3.1.18 CCSIDR_EL1, Cache Size ID Register, EL1.....	108
3.1.19 CLIDR_EL1, Cache Level ID Register, EL1.....	110
3.1.20 CPACR_EL1, Architectural Feature Access Control Register, EL1.....	112
3.1.21 CPTR_EL2, Architectural Feature Trap Register, EL2.....	113
3.1.22 CPTR_EL3, Architectural Feature Trap Register, EL3.....	117
3.1.23 CPUACTLR_EL1, CPU Auxiliary Control Register, EL1.....	118
3.1.24 CPUCFR_EL1, CPU Configuration Register, EL1.....	120
3.1.25 CPUECTLR_EL1, CPU Extended Control Register, EL1.....	121
3.1.26 CPUPCR_EL3, CPU Private Control Register, EL3.....	125
3.1.27 CPUPMR_EL3, CPU Private Mask Register, EL3.....	127
3.1.28 CPUPOR_EL3, CPU Private Operation Register, EL3.....	128
3.1.29 CPUPSELR_EL3, CPU Private Selection Register, EL3.....	129
3.1.30 CPUPWRCTLR_EL1, Power Control Register, EL1.....	131
3.1.31 CSSELR_EL1, Cache Size Selection Register, EL1.....	133
3.1.32 CTR_EL0, Cache Type Register, EL0.....	135

3.1.33 DCZID_EL0, Data Cache Zero ID Register, EL0.....	136
3.1.34 DISR_EL1, Deferred Interrupt Status Register, EL1.....	137
3.1.35 ERRIDR_EL1, Error ID Register, EL1.....	139
3.1.36 ERRSELR_EL1, Error Record Select Register, EL1.....	139
3.1.37 ERXCTLR_EL1, Selected Error Record Control Register, EL1.....	140
3.1.38 ERXFR_EL1, Selected Error Record Feature Register, EL1.....	140
3.1.39 ERXMISCO_EL1, Selected Error Record Miscellaneous Register 0, EL1.....	141
3.1.40 ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1....	141
3.1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1.....	142
3.1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1.....	144
3.1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1.....	145
3.1.44 ESR_EL1, Exception Syndrome Register, EL1.....	145
3.1.45 ESR_EL2, Exception Syndrome Register, EL2.....	146
3.1.46 ESR_EL3, Exception Syndrome Register, EL3.....	147
3.1.47 HACR_EL2, Hyp Auxiliary Configuration Register, EL2.....	148
3.1.48 HCR_EL2, Hypervisor Configuration Register, EL2.....	149
3.1.49 ID_AA64AFRO_EL1, AArch64 Auxiliary Feature Register 0.....	150
3.1.50 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1.....	150
3.1.51 ID_AA64DFRO_EL1, AArch64 Debug Feature Register 0, EL1.....	151
3.1.52 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1, EL1.....	152
3.1.53 ID_AA64ISARO_EL1, AArch64 Instruction Set Attribute Register 0, EL1.....	152
3.1.54 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1.....	154
3.1.55 ID_AA64MMFRO_EL1, AArch64 Memory Model Feature Register 0, EL1.....	156
3.1.56 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1.....	157
3.1.57 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1.....	159
3.1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1.....	161
3.1.59 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1, EL1.....	163
3.1.60 LORC_EL1, LORegion Control Register, EL1.....	164
3.1.61 LORID_EL1, LORegion ID Register, EL1.....	165
3.1.62 LORN_EL1, LORegion Number Register, EL1.....	166
3.1.63 MDCCR_EL3, Monitor Debug Configuration Register, EL3.....	166
3.1.64 MIDR_EL1, Main ID Register, EL1.....	168
3.1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1.....	169
3.1.66 PAR_EL1, Physical Address Register, EL1.....	171
3.1.67 REVIDR_EL1, Revision ID Register, EL1.....	172
3.1.68 RMR_EL3, Reset Management Register.....	172



3.1.69 RVBAR_EL3, Reset Vector Base Address Register, EL3.....	173
3.1.70 SCTLR_EL1, System Control Register, EL1.....	174
3.1.71 SCTLR_EL2, System Control Register, EL2.....	176
3.1.72 SCTLR_EL3, System Control Register, EL3.....	176
3.1.73 TCR_EL1, Translation Control Register, EL1.....	178
3.1.74 TCR_EL2, Translation Control Register, EL2.....	179
3.1.75 TCR_EL3, Translation Control Register, EL3.....	181
3.1.76 TTBR0_EL1, Translation Table Base Register 0, EL1.....	182
3.1.77 TTBR0_EL2, Translation Table Base Register 0, EL2.....	184
3.1.78 TTBR0_EL3, Translation Table Base Register 0, EL3.....	185
3.1.79 TTBR1_EL1, Translation Table Base Register 1, EL1.....	186
3.1.80 TTBR1_EL2, Translation Table Base Register 1, EL2.....	187
3.1.81 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2.....	187
3.1.82 VESR_EL2, Virtual SError Exception Syndrome Register.....	188
3.1.83 VTCR_EL2, Virtualization Translation Control Register, EL2.....	189
3.1.84 VTTBR_EL2, Virtualization Translation Table Base Register, EL2.....	190
3.2 Error System registers.....	191
3.2.1 Error System register summary.....	191
3.2.2 ERROCTL, Error Record Control Register.....	192
3.2.3 ERROFR, Error Record Feature Register.....	194
3.2.4 ERRORMISC0, Error Record Miscellaneous Register 0.....	196
3.2.5 ERROPFGCDN, Error Pseudo Fault Generation Count Down Register.....	198
3.2.6 ERROPFGCTL, Error Pseudo Fault Generation Control Register.....	198
3.2.7 ERROPFGF, Error Pseudo Fault Generation Feature Register.....	200
3.2.8 ERROSTATUS, Error Record Primary Status Register.....	202
3.3 GIC registers.....	205
3.3.1 CPU interface registers.....	206
3.3.2 AArch64 physical GIC CPU interface System register summary.....	206
3.3.3 ICC_APOR0_EL1, Interrupt Controller Active Priorities Group 0 Register 0, EL1.....	207
3.3.4 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Register 0 EL1.....	207
3.3.5 ICC_BPR0_EL1, Interrupt Controller Binary Point Register 0, EL1.....	208
3.3.6 ICC_BPR1_EL1, Interrupt Controller Binary Point Register 1, EL1.....	209
3.3.7 ICC_CTLR_EL1, Interrupt Controller Control Register, EL1.....	209
3.3.8 ICC_CTLR_EL3, Interrupt Controller Control Register, EL3.....	211
3.3.9 ICC_SRE_EL1, Interrupt Controller System Register Enable Register, EL1.....	214
3.3.10 ICC_SRE_EL2, Interrupt Controller System Register Enable register, EL2.....	215

3.3.11 ICC_SRE_EL3, Interrupt Controller System Register Enable register, EL3.....	216
3.3.12 AArch64 virtual GIC CPU interface register summary.....	218
3.3.13 ICV_APOR0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Register 0, EL1....	218
3.3.14 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Register 0, EL1....	219
3.3.15 ICV_BPR0_EL1, Interrupt Controller Virtual Binary Point Register 0, EL1.....	219
3.3.16 ICV_BPR1_EL1, Interrupt Controller Virtual Binary Point Register 1, EL1.....	220
3.3.17 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register, EL1.....	221
3.3.18 AArch64 virtual interface control System register summary.....	222
3.3.19 ICH_APOR0_EL2, Interrupt Controller Hyp Active Priorities Group 0 Register 0, EL2.....	223
3.3.20 ICH_AP1R0_EL2, Interrupt Controller Hyp Active Priorities Group 1 Register 0, EL2.....	223
3.3.21 ICH_HCR_EL2, Interrupt Controller Hyp Control Register, EL2.....	224
3.3.22 ICH_VMCR_EL2, Interrupt Controller Virtual Machine Control Register, EL2.....	227
3.3.23 ICH_VTR_EL2, Interrupt Controller VGIC Type Register, EL2.....	229
3.4 Advanced SIMD and floating-point registers.....	230
3.4.1 AArch64 register summary.....	230
3.4.2 FPCR, Floating-point Control Register.....	231
3.4.3 FPSR, Floating-point Status Register.....	233
<b>4 Debug descriptions.....</b>	<b>236</b>
4.1 Debug.....	236
4.1.1 About debug methods.....	236
4.1.2 Debug functional description.....	237
4.1.3 Debug register interfaces.....	240
4.1.4 Debug events.....	242
4.1.5 External debug interface.....	242
4.2 Performance Monitor Unit.....	243
4.2.1 About the PMU.....	243
4.2.2 PMU functional description.....	243
4.2.3 External register access permissions to the PMU registers.....	243
4.2.4 PMU events.....	244
4.2.5 PMU interrupts.....	258
4.2.6 Exporting PMU events.....	258
4.3 Embedded Trace Macrocell.....	259
4.3.1 About the ETM.....	259
4.3.2 ETM trace unit generation options and resources.....	259
4.3.3 ETM trace unit functional description.....	260

4.3.4 Resetting the ETM.....	262
4.3.5 Programming and reading ETM trace unit registers.....	262
4.3.6 ETM trace unit register interfaces.....	262
4.3.7 Interaction with the PMU and Debug.....	263
<b>5 Debug registers.....</b>	<b>264</b>
5.1 AArch64 debug registers.....	264
5.1.1 AArch64 Debug register summary.....	264
5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1.....	265
5.1.3 DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1.....	269
5.1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1.....	269
5.1.5 MDSCR_EL1, Monitor Debug System Control Register, EL1.....	272
5.2 Memory-mapped Debug registers.....	275
5.2.1 Memory-mapped Debug register summary.....	275
5.2.2 EDCIDR0, External Debug Component Identification Register 0.....	278
5.2.3 EDCIDR1, External Debug Component Identification Register 1.....	279
5.2.4 EDCIDR2, External Debug Component Identification Register 2.....	279
5.2.5 EDCIDR3, External Debug Component Identification Register 3.....	280
5.2.6 EDDEVID, External Debug Device ID Register 0.....	281
5.2.7 EDDEVID1, External Debug Device ID Register 1.....	282
5.2.8 EDDFR, External Debug Feature Register.....	282
5.2.9 EDITCTRL, External Debug Integration Mode Control Register.....	284
5.2.10 EDPFR, External Debug Processor Feature Register.....	284
5.2.11 EDPIDR0, External Debug Peripheral Identification Register 0.....	286
5.2.12 EDPIDR1, External Debug Peripheral Identification Register 1.....	287
5.2.13 EDPIDR2, External Debug Peripheral Identification Register 2.....	287
5.2.14 EDPIDR3, External Debug Peripheral Identification Register 3.....	288
5.2.15 EDPIDR4, External Debug Peripheral Identification Register 4.....	289
5.2.16 EDPIDRn, External Debug Peripheral Identification Registers 5-7.....	290
5.2.17 EDRCCR, External Debug Reserve Control Register.....	290
5.3 AArch64 PMU registers.....	291
5.3.1 AArch64 PMU register summary.....	291
5.3.2 PMCEID0_ELO, Performance Monitors Common Event Identification Register 0, ELO.....	293
5.3.3 PMCEID1_ELO, Performance Monitors Common Event Identification Register 1, ELO.....	296
5.3.4 PMCR_ELO, Performance Monitors Control Register, ELO.....	298
5.4 Memory-mapped PMU registers.....	301

5.4.1 Memory-mapped PMU register summary.....	301
5.4.2 PMCFGR, Performance Monitors Configuration Register.....	304
5.4.3 PMCIDR0, Performance Monitors Component Identification Register 0.....	305
5.4.4 PMCIDR1, Performance Monitors Component Identification Register 1.....	306
5.4.5 PMCIDR2, Performance Monitors Component Identification Register 2.....	306
5.4.6 PMCIDR3, Performance Monitors Component Identification Register 3.....	307
5.4.7 PMPIDR0, Performance Monitors Peripheral Identification Register 0.....	308
5.4.8 PMPIDR1, Performance Monitors Peripheral Identification Register 1.....	308
5.4.9 PMPIDR2, Performance Monitors Peripheral Identification Register 2.....	309
5.4.10 PMPIDR3, Performance Monitors Peripheral Identification Register 3.....	310
5.4.11 PMPIDR4, Performance Monitors Peripheral Identification Register 4.....	311
5.4.12 PMPIDRn, Performance Monitors Peripheral Identification Register 5-7.....	312
5.5 PMU snapshot registers.....	312
5.5.1 PMU snapshot register summary.....	312
5.5.2 PMPCSSR, PMU Snapshot Program Counter Sample Register.....	313
5.5.3 PMCIDSSR, PMU Snapshot CONTEXTIDR_EL1 Sample Register.....	314
5.5.4 PMCID2SSR, PMU Snapshot CONTEXTIDR_EL2 Sample Register.....	314
5.5.5 PMSSSR, PMU Snapshot Status Register.....	314
5.5.6 PMOVSSR, PMU Snapshot Overflow Status Register.....	315
5.5.7 PMCCNTSR, PMU Snapshot Cycle Counter Register.....	316
5.5.8 PMEVCNTSRn, PMU Snapshot Cycle Counter Registers 0-5.....	316
5.5.9 PMSSCR, PMU Snapshot Capture Register.....	316
5.6 ETM registers.....	317
5.6.1 ETM register summary.....	317
5.6.2 TRCACATRn, Address Comparator Access Type Registers 0-7.....	320
5.6.3 TRCACVRn, Address Comparator Value Registers 0-7.....	321
5.6.4 TRCAUTHSTATUS, Authentication Status Register.....	322
5.6.5 TRCAUXCTLR, Auxiliary Control Register.....	323
5.6.6 TRCBBCTLR, Branch Broadcast Control Register.....	325
5.6.7 TRCCCCTLR, Cycle Count Control Register.....	326
5.6.8 TRCCIDCCTLR0, Context ID Comparator Control Register 0.....	327
5.6.9 TRCCIDCVR0, Context ID Comparator Value Register 0.....	327
5.6.10 TRCCIDR0, ETM Component Identification Register 0.....	328
5.6.11 TRCCIDR1, ETM Component Identification Register 1.....	329
5.6.12 TRCCIDR2, ETM Component Identification Register 2.....	329
5.6.13 TRCCIDR3, ETM Component Identification Register 3.....	330

5.6.14 TRCCCLAIMCLR, Claim Tag Clear Register.....	331
5.6.15 TRCCCLAIMSET, Claim Tag Set Register.....	332
5.6.16 TRCCNTCTLR0, Counter Control Register 0.....	332
5.6.17 TRCCNTCTLR1, Counter Control Register 1.....	334
5.6.18 TRCCNTRLDVRn, Counter Reload Value Registers 0-1.....	336
5.6.19 TRCCNTVRn, Counter Value Registers 0-1.....	336
5.6.20 TRCCONFIGR, Trace Configuration Register.....	337
5.6.21 TRCDEVAFF0, Device Affinity Register 0.....	340
5.6.22 TRCDEVAFF1, Device Affinity Register 1.....	340
5.6.23 TRCDEVARCH, Device Architecture Register.....	340
5.6.24 TRCDEVID, Device ID Register.....	341
5.6.25 TRCDEVTYPE, Device Type Register.....	342
5.6.26 TRCEVENTCTL0R, Event Control 0 Register.....	342
5.6.27 TRCEVENTCTL1R, Event Control 1 Register.....	344
5.6.28 TRCEXTINSELR, External Input Select Register.....	346
5.6.29 TRCIDR0, ID Register 0.....	347
5.6.30 TRCIDR1, ID Register 1.....	349
5.6.31 TRCIDR2, ID Register 2.....	350
5.6.32 TRCIDR3, ID Register 3.....	351
5.6.33 TRCIDR4, ID Register 4.....	353
5.6.34 TRCIDR5, ID Register 5.....	355
5.6.35 TRCIDR8, ID Register 8.....	356
5.6.36 TRCIDR9, ID Register 9.....	357
5.6.37 TRCIDR10, ID Register 10.....	357
5.6.38 TRCIDR11, ID Register 11.....	358
5.6.39 TRCIDR12, ID Register 12.....	358
5.6.40 TRCIDR13, ID Register 13.....	359
5.6.41 TRCIMSPEC0, Implementation Specific Register 0.....	359
5.6.42 TRCITATBIDR, Integration ATB Identification Register.....	360
5.6.43 TRCITCTRL, Integration Mode Control Register.....	361
5.6.44 TRCITIATBINR, Integration Instruction ATB In Register.....	361
5.6.45 TRCITIATBOUTr, Integration Instruction ATB Out Register.....	362
5.6.46 TRCITIDATAR, Integration Instruction ATB Data Register.....	363
5.6.47 TRCLAR, Software Lock Access Register.....	364
5.6.48 TRCLSR, Software Lock Status Register.....	365
5.6.49 TRCOSLAR, OS Lock Access Register.....	365

5.6.50 TRCOSLSR, OS Lock Status Register.....	366
5.6.51 TRCPDCR, Power Down Control Register.....	367
5.6.52 TRCPDSR, Power Down Status Register.....	368
5.6.53 TRCPIDR0, ETM Peripheral Identification Register 0.....	369
5.6.54 TRCPIDR1, ETM Peripheral Identification Register 1.....	369
5.6.55 TRCPIDR2, ETM Peripheral Identification Register 2.....	370
5.6.56 TRCPIDR3, ETM Peripheral Identification Register 3.....	371
5.6.57 TRCPIDR4, ETM Peripheral Identification Register 4.....	372
5.6.58 TRCPIDRn, ETM Peripheral Identification Registers 5-7.....	372
5.6.59 TRCPRGCTLR, Programming Control Register.....	373
5.6.60 TRCRSCTLRn, Resource Selection Control Registers 2-15.....	373
5.6.61 TRCSEQEVRn, Sequencer State Transition Control Registers 0-2.....	375
5.6.62 TRCSEQRSTEVr, Sequencer Reset Control Register.....	376
5.6.63 TRCSEQSTR, Sequencer State Register.....	377
5.6.64 TRCSSCCR0, Single-Shot Comparator Control Register 0.....	378
5.6.65 TRCSSCSR0, Single-Shot Comparator Status Register 0.....	379
5.6.66 TRCSTALLCTLR, Stall Control Register.....	380
5.6.67 TRCSTATR, Status Register.....	381
5.6.68 TRCSYNCPR, Synchronization Period Register.....	382
5.6.69 TRCTRACEIDR, Trace ID Register.....	382
5.6.70 TRCTSCTLR, Global Timestamp Control Register.....	383
5.6.71 TRCVICTLR, ViewInst Main Control Register.....	384
5.6.72 TRCVIIECTLR, ViewInst Include-Exclude Control Register.....	386
5.6.73 TRCVISSCTLR, ViewInst Start-Stop Control Register.....	387
5.6.74 TRCVMIDCCTLRO, Virtual context identifier Comparator Control Register 0.....	388
5.6.75 TRCVMIDCVRO, VMID Comparator Value Register 0.....	389
<b>A Revisions.....</b>	<b>390</b>
A.1 Revisions.....	390

# 1 Introduction

## 1.1 Product revision status

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

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

## 1.2 Intended audience

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

## 1.3 Conventions

The following subsections describe conventions used in Arm documents.







### Glossary

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

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

### Typographic conventions

Convention	Use
<i>italic</i>	Citations.
<b>bold</b>	Interface elements, such as menu names.  Signal names.  Terms in descriptive lists, where appropriate.
monospace	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
<b>monospace bold</b>	Language keywords when used outside example code.
monospace <u>underline</u>	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

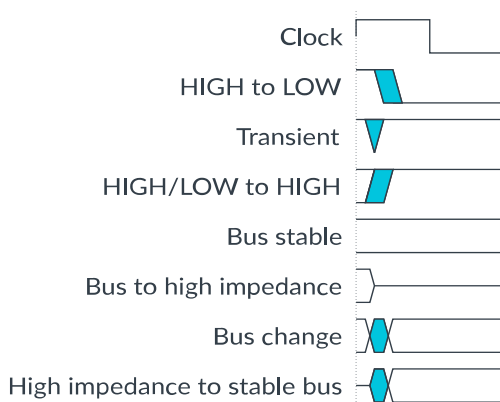
Convention	Use
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments.  For example:  <pre>MRC p15, 0, &lt;Rd&gt;, &lt;CRn&gt;, &lt;CRm&gt;, &lt;Opcode_2&gt;</pre>
<b>SMALL CAPITALS</b>	Terms that have specific technical meanings as defined in the <i>Arm® Glossary</i> . For example, <b>IMPLEMENTATION DEFINED</b> , <b>IMPLEMENTATION SPECIFIC</b> , <b>UNKNOWN</b> , and <b>UNPREDICTABLE</b> .
 Caution	Recommendations. Not following these recommendations might lead to system failure or damage.
 Warning	Requirements for the system. Not following these requirements might result in system failure or damage.
 Danger	Requirements for the system. Not following these requirements will result in system failure or damage.
 Note	An important piece of information that needs your attention.
 Tip	A useful tip that might make it easier, better or faster to perform a task.
 Remember	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
Arm® Architecture Reference Manual Armv8, for A-profile architecture	DDI 0487	No
Arm® Cortex®-A65AE Core Cryptographic Extension Technical Reference Manual	101388	No
Arm® DynamIQ™ Shared Unit AE Technical Reference Manual	101322	No
Arm® CoreSight™ Architecture Specification v3.0	IHI 0029	No
Arm® CoreSight™ ELA-500 Embedded Logic Analyzer Technical Reference Manual	100127	No
AMBA® AXI and ACE Protocol Specification	IHI 0022	No
AMBA® APB Protocol Version 2.0 Specification	IHI 0024	No
AMBA® 5 CHI Architecture Specification	IHI 0050	No
AMBA® ATB Protocol Specification	IHI 0032	No
AMBA® Low Power Interface Specification Arm® Q-Channel and P-Channel Interfaces	IHI 0068	No
Arm® Debug Interface Architecture Specification, ADIV5.0 to ADIV5.2	IHI 0031	No
Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4	IHI 0069	No
Arm® Embedded Trace Macrocell Architecture Specification ETMv4	IHI 0064	No
Arm® Reliability, Availability, and Serviceability (RAS) Specification, Armv8, for the Armv8-A architecture profile	DDI 0587	No
Arm® Cortex®-A65AE Core Configuration and Sign-off Guide	101386	Yes
Arm® Cortex®-A65AE Core Integration Manual	101387	Yes
Arm® DynamIQ™ Shared Unit AE Configuration and Sign-off Guide	101323	Yes
Arm® DynamIQ™ Shared Unit AE Integration Manual	101324	Yes

**Table 1-3: Other publications**

Document ID	Organization	Document name
-	-	ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic



Arm® floating-point terminology is largely based on the earlier ANSI/IEEE Std 754-1985 issue of the standard. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information.



Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at <http://www.adobe.com>

---

## 2 Functional description

This part describes the main functionality of the Cortex®-A65AE core.

### 2.1 Introduction

This chapter provides an overview of the Cortex®-A65AE core and its features.

#### 2.1.1 About the core

The Cortex®-A65AE core is a power-efficient, mid-range, throughput computing oriented, *Simultaneously MultiThreaded* (SMT) core that implements the AArch64 Execution state of the Arm®v8-A architecture inside the *DynamlQ Shared Unit AE* (DSU-AE) cluster.

The Cortex®-A65AE core supports three DSU-AE execution modes:

- Split-mode, where the cores in each core pair execute independently of each other.
- Lock-mode, where one of the cores in a core pair functions as a redundant copy of the primary function core.
- Hybrid-mode, where the cores execute independently, as in Split-mode, while the DSU-AE executes in lock-step, as in Lock-mode.



When running in Lock-mode, a *core pair* is defined as a pair of cores that are viewed architecturally by the DSU-AE as a single core. For more information on the DSU-AE and running in Split-mode, Lock-mode, or Hybrid-mode, see *Functional description* in the *Arm® DynamlQ™ Shared Unit AE Technical Reference Manual*.

---

The Cortex®-A65AE core supports:

- The Arm®v8.2-A extension.
- The *Reliability, Availability, and Serviceability* (RAS) extension
- The Load acquire (LDABR) instructions introduced in the Arm®v8.3-A extension
- The Dot Product support instructions introduced in the Arm®v8.4-A extension.
- The PSTATE *Speculative Store Bypass Safe* (SSBS) bit that supports software mitigation for Spectre Variant 4 introduced in the Arm®v8.5-A extension.

The Cortex®-A65AE core does not implement the AArch32 Execution state of the Arm®v8-A architecture.

As an SMT core, the Cortex®-A65AE core supports two execution threads on each core. Each thread is a separate architectural *Processing Element* (PE), and so has a complete copy of the architectural state.

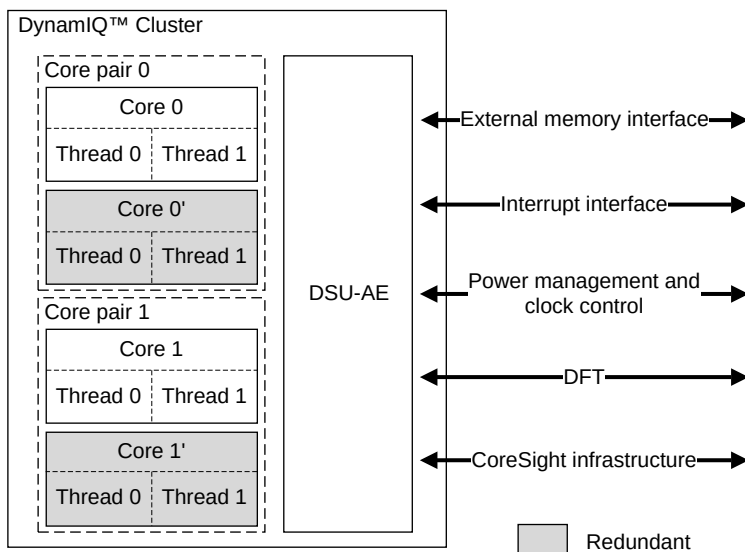
The simultaneous nature of the multithreading means that the Cortex®-A65AE core is able to issue and execute instructions from both threads simultaneously in the same cycle. Software views a thread on the Cortex®-A65AE core in the same way as it would view a core in a traditional, single-threaded, multi-core processor. Thus, existing software for both single core and multi-core can run unmodified on the Cortex®-A65AE core.

In a DSU-AE cluster, each Cortex®-A65AE core has a separate *Level 1* (L1) instruction cache, a separate L1 data cache, and an optional private per-core *Level 2* (L2) cache. For more information, see the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual*.

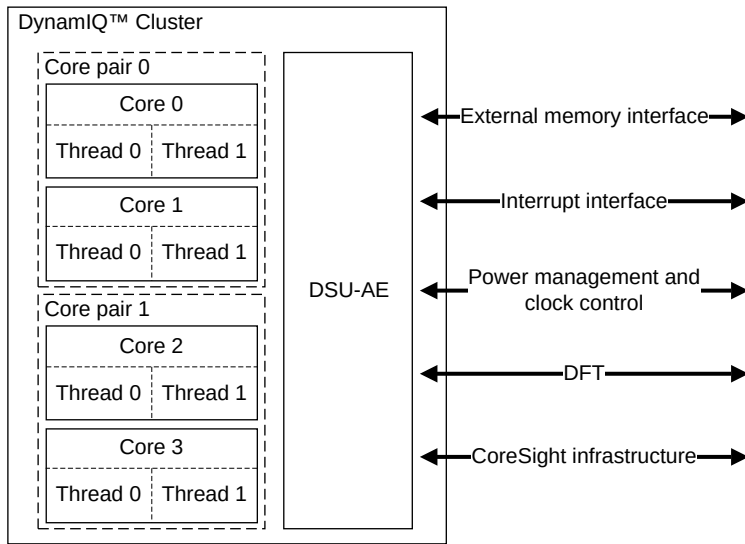
The Cortex®-A65AE core cannot be instantiated as a single core. The Cortex®-A65AE core must be used in a core pair configuration with a maximum of four core pairs in each cluster, for a total of eight cores.

The following figure shows an example of two Cortex®-A65AE core pairs in Lock-mode.

**Figure 2-1: Example Cortex®-A65AE core configuration with two core pairs in Lock-mode**



The following figure shows an example of two Cortex®-A65AE core pairs in Split-mode.

**Figure 2-2: Example Cortex®-A65AE core configuration with two core pairs in Split-mode**

## 2.1.2 Features

The Cortex®-A65AE core includes the following features:

### Core features

- Full implementation of the Arm®v8.2-A A64 instruction set.
- AArch64 Execution state at all Exception levels (EL0 to EL3).
- Superscalar, variable-length, out-of-order pipeline.
- Simultaneous multithreading, with two execution threads on each core.
- *Memory Management Unit* (MMU).
- Support for Arm *TrustZone*® technology.
- 44-bit physical address (PA). Optional execution unit that implements the Advanced SIMD and floating-point architecture support.
- Optional Cryptographic Extension. This architectural extension is only available if the Advanced SIMD and floating-point execution unit is present.
- *Generic Interrupt Controller* (GIC) CPU interface to connect to an external distributor.
- Generic Timer interface supporting 64-bit count input from an external system counter.
- Implementation of the Split-mode, Lock-mode, or Hybrid-mode with the ability to choose a mode based on the *DynamIQ Shared Unit AE* (DSU-AE) *CEMODE* input signal during cluster Cold reset.

### Cache features

- Separate L1 data and instruction caches.

- Optional unified private L2 cache.
- L1 and L2 cache protection in the form of *Error Correcting Code* (ECC) or parity cache protection on all RAM instances.

#### Debug features

- *Reliability, Availability, and Serviceability* (RAS) Extension.
- Arm®v8.2-A debug logic.
- *Performance Monitor Unit* (PMU).
- *Embedded Trace Macrocell* (ETM) that supports instruction trace only.

### 2.1.3 Split-Lock

The *DynamlQ Shared Unit* AE (DSU-AE) provides a boot-time option for the cluster to execute in either Split-mode, Lock-mode, or Hybrid-mode.

The Split-mode, Lock-mode, and Hybrid-mode extend the functionality of a typical *Dual-Core Lock-Step* (DCLS) system by changing its execution mode at reset. For instance, while some modes enable more logical cores, others provide core redundancy. The potential of core redundancy requires an even number of cores in the DSU-AE cluster.



The DSU-AE **CEMODE** input signal determines whether the cluster enters the Lock-mode, Split-mode, or Hybrid-mode at reset.

In Split-mode, cores execute independently. The DCLS-related comparators, timeout detectors, and redundant DSU-AE logic are clock gated and idle. Each core has its own independent clock. As a result, each core can be powered down independently.

In Lock-mode, one of the cores in a core pair functions as a redundant copy of the primary function core. A core pair is defined as a pair of cores that are viewed architecturally as a single core when executing in Lock-mode. The same inputs drive both the primary logic and the redundant logic, and the redundant core executes in lock-step with the primary function core. Therefore, both cores in a core pair must be of identical configuration with the same microarchitecture and configuration parameters.

Furthermore, in Lock-mode, the DSU-AE cluster utilizes redundant logic to execute in lock-step with the primary logic. For this reason, the entire cluster is executing as a DCLS system. The primary logic drives the outputs to the system, although these outputs are compared with the redundant logic. Any divergence is reported to the system. If a fault is detected, both of the cores are permitted to continue execution but the results are **UNPREDICTABLE**.

Hybrid-mode is a mixed execution mode where the cores execute independently, as in Split-mode, while the DSU-AE executes in lock-step, as in Lock-mode. Therefore, in Hybrid-mode, the DSU-AE cluster provides a partial DCLS solution with the following benefits:

- Compared to Lock-mode, Hybrid-mode offers better cluster performance, since the cores execute independently of each other.
- Compared to Split-mode, Hybrid-mode offers better cluster fault tolerance, since the DSU-AE executes in lock-step.



From a system or software perspective, there is no difference between the Split-mode and Hybrid-mode.

For more information on Split-mode, Lock-mode, and Hybrid-mode, see *Functional description* in the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual*.

## 2.1.4 Implementation options

The Cortex®-A65AE core is highly configurable.

Build-time configuration options make it possible to meet functional requirements with the smallest possible area and power. All cores in a cluster have the same build-time configuration except for the L2 cache inclusion and size. In Lock-mode, both cores in a core pair must also have the same L2 cache size. For example, core 0 and core 1 in core pair 0 must have the same L2 cache size parameter.

The following table lists the implementation options for a core.



The features implemented are shared resources for both threads. The selected cache options are dynamically shared per usage.

**Table 2-1: Implementation options for a core**

Feature	Range of options	Notes
L1 instruction cache size	<ul style="list-style-type: none"> <li>• 32KB</li> <li>• 64KB</li> </ul>	-
L1 data cache size	<ul style="list-style-type: none"> <li>• 32KB</li> <li>• 64KB</li> </ul>	-
L2 cache	<ul style="list-style-type: none"> <li>• Included</li> <li>• Not included</li> </ul>	-
L2 cache size	<ul style="list-style-type: none"> <li>• 64KB</li> <li>• 128KB</li> <li>• 256KB</li> </ul>	-
Advanced SIMD and floating-point support	<ul style="list-style-type: none"> <li>• Included</li> <li>• Not included</li> </ul>	There is no option to implement floating-point without Advanced SIMD.

Feature	Range of options	Notes
Cryptographic Extension	<ul style="list-style-type: none"> <li>Included</li> <li>Not included</li> </ul>	Requires Advanced SIMD and floating-point support.
CoreSight <i>Embedded Logic Analyzer</i> (ELA)	<ul style="list-style-type: none"> <li>Optional support</li> </ul>	Support for integrating CoreSight ELA-500. CoreSight is a range of debug components available as a separately licensable product.
Dot Product support instructions	<ul style="list-style-type: none"> <li>Included</li> </ul>	Support for the Arm®v8.4-A <code>SDOT</code> and <code>UDOT</code> instructions. Requires Advanced SIMD and floating-point support.

## 2.1.5 Supported standards and specifications

The Cortex®-A65AE core implements the Arm®v8-A architecture and some architecture extensions. It also supports various interconnect, interrupt, timer, debug, and trace architectures.

**Table 2-2: Compliance with standards and specifications**

Architecture specification or standard	Version	Notes
Arm architecture	Arm®v8-A	<ul style="list-style-type: none"> <li>AArch64 Execution state only</li> <li>All Exception levels, EL0-EL3</li> <li>A64 instruction set</li> </ul>
Arm architecture extensions	<ul style="list-style-type: none"> <li>Arm®v8.1-A extensions</li> <li>Arm®v8.2-A extensions</li> <li>Advanced SIMD and floating-point support</li> <li>Cryptographic Extension</li> <li>RAS Extension</li> <li>Arm®v8.3-A LDAPR instructions</li> <li>Arm®v8.4-A Dot Product support instructions</li> <li>Arm®v8.5-A extensions</li> </ul>	<ul style="list-style-type: none"> <li>You cannot implement floating-point without Advanced SIMD.</li> <li>You cannot implement the Cryptographic Extension without the Advanced SIMD and floating-point support.</li> <li>The Cortex®-A65AE core implements the LDAPR instructions introduced in the Arm®v8.3-A extensions.</li> <li>From the Arm®v8.4-A extensions, the Cortex®-A65AE core only implements the <code>UDOT</code> and <code>SDOT</code> instructions.</li> <li>The Cortex®-A65AE core implements the PSTATE SSBS (Speculative Store Bypass Safe) bit that supports software mitigation for Spectre Variant 4 introduced in the Arm®v8.5-A extension.</li> </ul>
Generic Interrupt Controller	GICv4	Backwards compatible with GICv3  <b>Note:</b> The <i>DynamlQ Shared Unit AE</i> (DSU-AE) uses Affinity Level 1 to distinguish between different cores, which some interrupt controllers, such as GIC-500, do not support.
Performance Monitor Unit	PMUv3	-
Debug	Arm®v8-A	With support for the debug features added by the Arm®v8.2-A extensions



Architecture specification or standard	Version	Notes
CoreSight	CoreSightv3	-
Embedded Trace Macrocell	ETMv4.2	<ul style="list-style-type: none"> <li>• Instruction trace only. Data trace is not supported.</li> <li>• System register access to the ETM is not supported.</li> </ul>

See [Additional Reading HeliosAE TRM](#) for a list of architectural references.

## 2.1.6 Test features

The Cortex®-A65AE 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 processing logic and memory arrays.

For more information, see the *Arm® Cortex®-A65AE Core Integration Manual*.

## 2.1.7 Design tasks

The Cortex®-A65AE core is delivered as a synthesizable *Register Transfer Level* (RTL) description in Verilog HDL. Before you can use the Cortex®-A65AE core, you must implement it, integrate it, and program it.

A different party can perform each of the following tasks. Each task can include implementation and integration choices that 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 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 following:

### Build configuration

The implementer chooses the options that affect how the RTL source files are pre-processed. These options usually include or exclude logic that affects one or more of 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 particular values into registers. The configuration choices affect the behavior of the core.

### 2.1.8 Product revisions

This section indicates the first release and, in subsequent releases, describes the differences in functionality between product revisions.

r0p0	First release.
r1p0	Support for the following changes made to the <i>DynamiQ Shared Unit AE</i> (DSU-AE) R1: <ul style="list-style-type: none"><li>• Configurable temporal diversity</li><li>• Hybrid-mode</li></ul>
r1p1	Errata fixes only. No features added from previous release.

For more information, see [A.1 Revisions](#) on page 390.

## 2.2 Technical overview

This chapter describes the structure of the Cortex®-A65AE core.

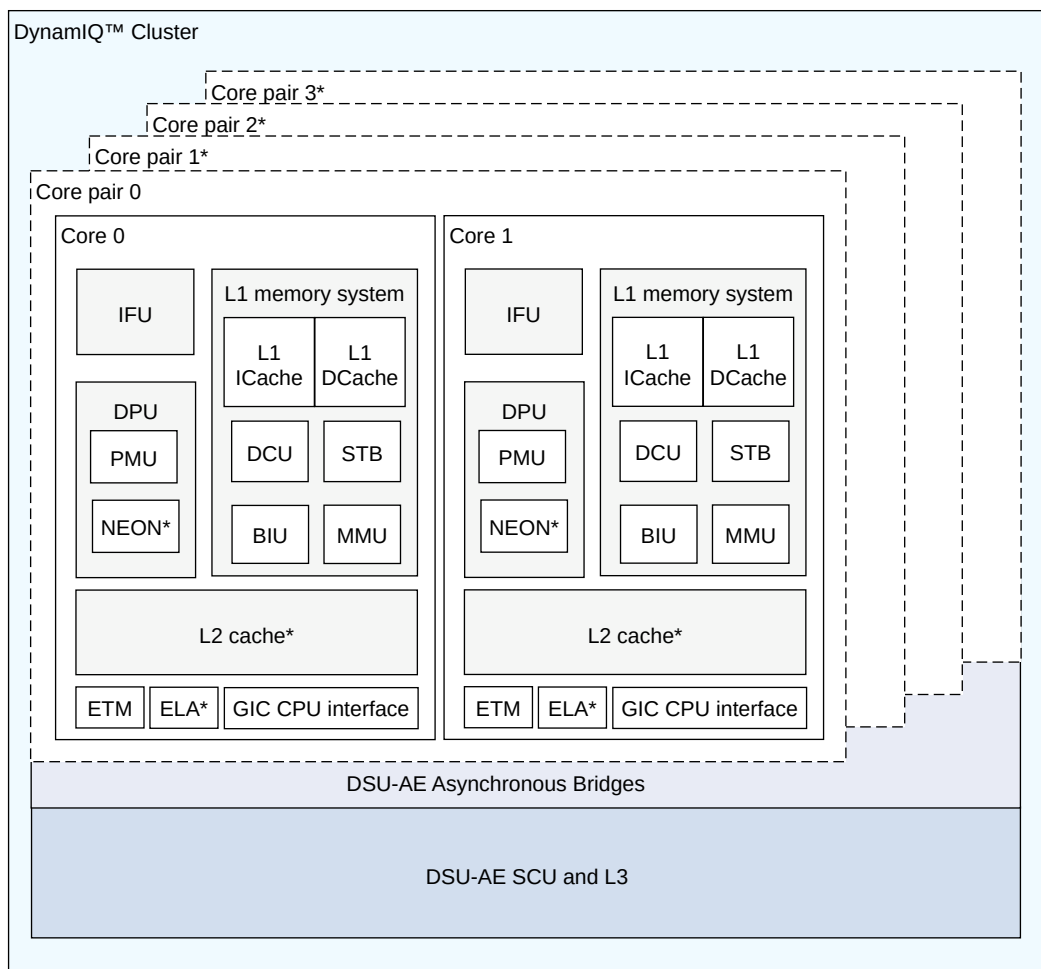
### 2.2.1 Components

The cluster consists of:

- One to four core pairs.
- The DSU-AE, which connects the cores to an external memory system.

For more information, see the *Arm® DynamiQ™ Shared Unit AE Technical Reference Manual*.

The following figure includes a top-level functional diagram of a core pair in Split-mode. In Lock-mode, Core 1 is named Core 0'.

**Figure 2-3: Core pair block diagram**

\* Optional



There are multiple asynchronous bridges between the Cortex®-A65AE core and the DSU-AE. Only the coherent interface between the Cortex®-A65AE core and the DSU-AE can be configured to run synchronously, however it does not affect the other interfaces such as debug, trace, and GIC which are always asynchronous. For more information on how to set the coherent interface to run either synchronously or asynchronously, see *Configuration Guidelines* in the *Arm® DynamIQ™ Shared Unit AE Configuration and Sign-off Guide*.

### Instruction Fetch Unit (IFU)

The IFU fetches instructions from the instruction cache or from external memory and predicts the outcome of branches in the instruction stream. It passes the instructions to the *Data Processing Unit* (DPU) for processing.

## Data Processing Unit (DPU)

The DPU decodes and executes instructions. It executes instructions that require data transfer to or from the memory system by interfacing to the *Data Cache Unit* (DCU). The DPU includes the PMU, the Advanced SIMD and floating-point support, and the Cryptographic Extension.

### PMU

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.

### Advanced SIMD and floating-point support

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

All scalar floating-point instructions are available in the A64 instruction set.

The A64 instruction set offers additional Advanced SIMD instructions, including double-precision floating-point vector operations.



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

---

## Cryptographic Extension

The optional Cortex®-A65AE core Cryptographic Extension supports the Arm®v8-A Cryptographic Extension. It is a configuration option that can be set when configuring and integrating the core into a system and applies to all cores. The Cryptographic Extension adds new instructions to Advanced SIMD that accelerate:

- *Advanced Encryption Standard* (AES) encryption and decryption.
- The *Secure Hash Algorithm* (SHA) functions SHA-1, SHA-224, and SHA-256.
- Finite field arithmetic used in algorithms such as *Galois/Counter Mode* and *Elliptic Curve Cryptography*.

## Memory Management Unit (MMU)

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

### L1 TLBs

The first level of caching for the translation table information is an L1 TLB. It is implemented on both of the instruction and data sides. All TLB-related maintenance operations result in flushing both the instruction and data L1 TLBs.

## L2 TLB

A unified L2 TLB handles the misses from the L1 TLBs.

Parity bits protect the TLB RAMs by enabling the detection of any single-bit error. If an error is detected, the entry is invalidated and fetched again.

## L1 memory system

The L1 memory system includes the *Data Cache Unit* (DCU), the *Store Buffer* (STB), and the *Bus Interface Unit* (BIU).

### DCU

The DCU manages all load and store operations.

The L1 data cache RAMs are protected using *Error Correction Codes* (ECC). The ECC scheme is *Single Error Correct Double Error Detect* (SECCDED). The DCU includes a combined local and global exclusive monitor that is used by Load-Exclusive and Store-Exclusive instructions.

### STB

The STB holds store operations when they have left the load/store pipeline in the DCU and have been committed by the DPU. The STB can request access to the L1 data cache, initiate linefills, or write to L2 and L3 memory systems.

The STB is also used to queue maintenance operations before they are broadcast to other cores in the cluster.

### BIU

The BIU contains the interface to the L2 memory system and buffers to decouple the interface from the L1 data cache and STB.

## L2 memory system

The L2 memory system contains the L2 cache. The L2 cache is optional and private to each core. The L2 cache is 4-way set associative, supports 64-byte cache lines, and has a configurable cache RAM size between 64KB and 256KB. The L2 memory system is connected to the DynamIQ™ Shared Unit AE through an optional asynchronous bridge.

## GIC CPU interface

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

## DynamIQ™ Shared Unit AE

The DSU-AE contains the L3 cache and logic required to maintain coherence between the cores in the cluster. For more information, see the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual*.

## Debug and trace components

The Cortex®-A65AE core supports a range of debug, test, and trace options including:

- Six performance event counters, provided by the PMU, and one cycle counter.

- Six hardware breakpoints, and four watchpoints.
- Per-core instruction trace only ETM.
- Per-core support for an ELA-500.
- AMBA 4 APB interfaces between the cluster and the DebugBlock.

Details of the core-specific debug elements can be found in this document. For information on the cluster debug and trace components supported by the Cortex®-A65AE core, see the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual*.

### Related information

[Memory Management Unit](#) on page 47

[L1 memory system](#) on page 56

[L2 memory system](#) on page 72

[Generic Interrupt Controller CPU interface](#) on page 81

[Debug](#) on page 236

[Performance Monitor Unit](#) on page 242

[Embedded Trace Macrocell](#) on page 258

## 2.2.2 Interfaces

The Cortex®-A65AE core has several interfaces to connect it to a SoC. The *DynamIQ Shared Unit AE* (DSU-AE) manages all interfaces.

For information on the interfaces, see the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual*.

## 2.2.3 About 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:

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

Some of the System registers are accessible through the external debug interface.

## 2.2.4 About the Generic Timer

The Generic Timer can schedule events and trigger interrupts that are based on an incrementing counter value. It generates timer events as active-LOW interrupt outputs and event streams.

The Cortex®-A65AE core provides a set of timer registers for each thread. The timers are:

- An EL1 Non-secure physical timer
- An EL2 Hypervisor physical timer
- An EL3 Secure physical timer
- A virtual timer
- A Hypervisor virtual timer

The Cortex®-A65AE core does not include the system counter. The system counter resides in the SoC, and its value is distributed to the core over a 64-bit bus.

For more information on the Generic Timer, see the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual* and the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 2.3 Clocks, resets, and input synchronization

This chapter describes the clocks, resets, and input synchronization of the Cortex®-A65AE core.

### 2.3.1 About clocks, resets, and input synchronization

The Cortex®-A65AE core supports hierarchical clock gating.

The Cortex®-A65AE core contains several interfaces that connect to other components in the system. These interfaces can be in the same clock domain or in other clock domains.

For information about clocks, resets, and input synchronization, see the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual*.

### 2.3.2 Asynchronous interface

Your implementation can include an optional asynchronous interface between the core and the *DynamIQ Shared Unit AE* (DSU-AE) top level.

See *Implementation options* in the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual* for more information.

## 2.4 Power management

This chapter describes the power domains and the power modes in the Cortex®-A65AE core.

### 2.4.1 About power management

The Cortex®-A65AE core provides mechanisms to control both dynamic and static power dissipation.

The dynamic power management includes the following features:

- Architectural clock gating.
- Per-core *Dynamic Voltage Frequency Scaling* (DVFS).



DVFS temporarily gates the source clock to lock a new frequency. If the source clock is gated too long, it can lead to a false divergence being reported in Lock-mode. In Lock-mode, DVFS is only safe and allowed when:

- The targeted clock domain is idle.
- The core pair is either in retention mode or Off mode.

The static power management includes the following features:

- Powerdown.
- Dynamic core retention schemes to lower the voltage of a core pair.

#### Related information

[Power domains](#) on page 33

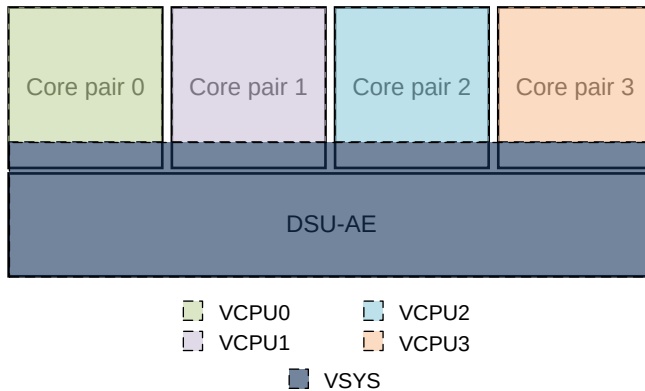
[Power control](#) on page 38

### 2.4.2 Voltage domains

The Cortex®-A65AE core supports a VCPU voltage domain and a VSYS voltage domain.

The following figure shows the VCPU and VSYS voltage domains in each Cortex®-A65AE core pair and in the DSU-AE. The example shows a configuration with four Cortex®-A65AE core pairs, where each core pair is driven by the same voltage domain.



**Figure 2-4: Cortex®-A65AE voltage domains**

Asynchronous bridge logic exists between the voltage domains. The Cortex®-A65AE core processing logic and core clock domain of the asynchronous bridge are in the VCPU voltage domain. The DSU-AE clock domain of the asynchronous bridge is in the VSYS voltage domain.



You can tie VCPU and VSYS to the same supply if the core pair is not required to support *Dynamic Voltage and Frequency Scaling* (DVFS).

In Lock-mode, the core pair supports DVFS, and each core in a core pair has the same voltage domain and clock domain.

In Split-mode, each core in a core pair supports independent *Dynamic Frequency Scaling* (DFS), and each core in a core pair has a separate power domain. This separate power domain allows each core in a core pair to be powered down independently.

### 2.4.3 Power domains

The Cortex®-A65AE core contains a core power domain (PDCPU), an Advanced SIMD and floating-point power domain (PDADVSIMD), and a core top-level SYS power domain (PDSYS).

The PDCPU power domain contains all core processing logic excluding the cluster clock domain side of the bridge.

The PDADVSIMD power domain is a separate power domain to support the Advanced SIMD and floating-point block. To support independent retention control, each Advanced SIMD and floating-point block also have its own power domain for isolation from the surrounding domain.

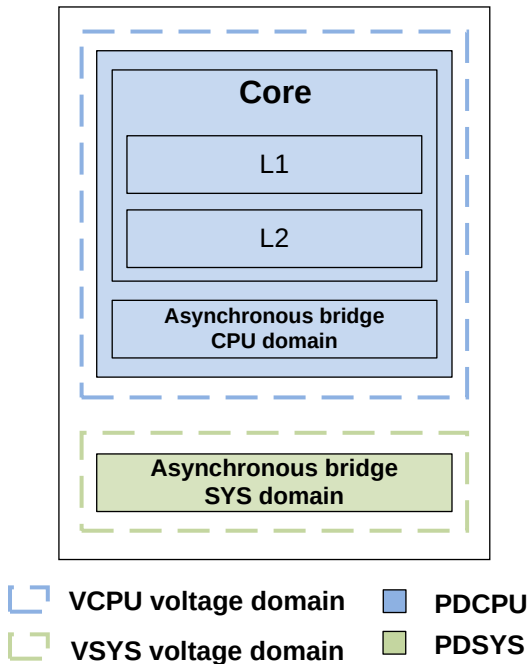
The PDSYS power domain contains the cluster clock domain side of the bridge.



There are additional system power domains in the *DynamiQ Shared Unit AE* (DSU-AE). See the *Arm® DynamiQ™ Shared Unit AE Technical Reference Manual* for information.

The following figure shows an example of how the voltage and power domains are organized.

**Figure 2-5: Cortex®-A65AE core power domain diagram**



The following table describes the power domains that the Cortex®-A65AE core supports.

**Table 2-3: Power domain description**

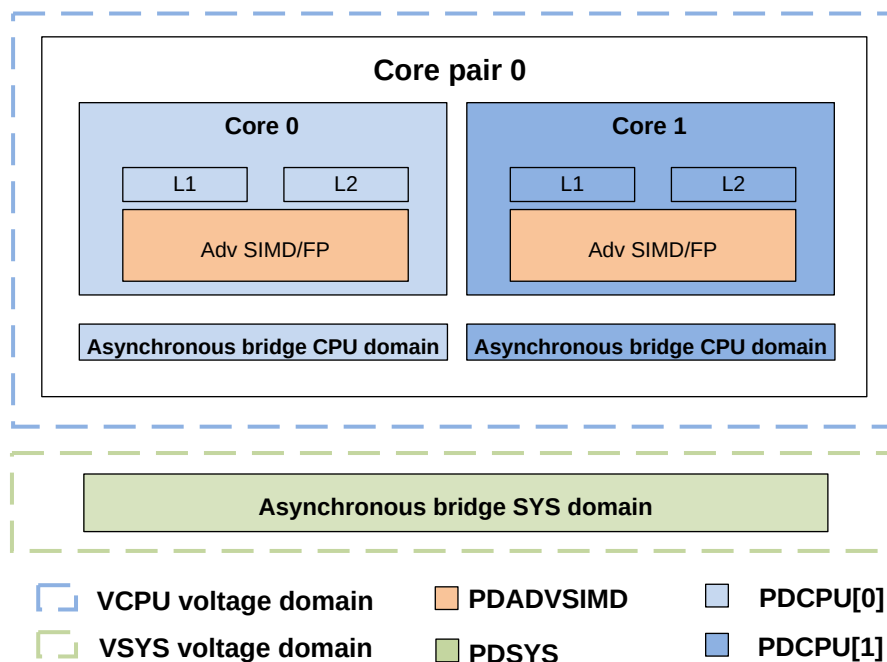
Power domain	Description
PDCPU<n>	<p>The domain includes the L1 and L2 TLBs, L1 and L2 cache RAMs, the Debug registers, and the core asynchronous bridge logic within the <code>u_vcpu</code> hierarchy that corresponds to this core. If a separate domain is not included for the Advanced SIMD and floating-point block, then this block is included in the PDCPU&lt;n&gt; domain.</p> <p>&lt;n&gt; is the core number in the range 0-7. The number represents core 0, core 1, core 2, to core 7. If a core is not present, the corresponding power domain is not present.</p>
PDADVSIMD<n>	<p>An optional power domain for Advanced SIMD and floating-point block to implement SIMD dynamic retention.</p> <p>&lt;n&gt; is the core number in the range 0-7. The number represents core 0, core 1, core 2, to core 7. If a core is not present, the corresponding power domain is not present.</p>

Power domain	Description
PDSYS	The domain is the interface between the Cortex®-A65AE and the DSU-AE. It contains the cluster clock domain logic of the CPU bridge. The CPU bridge contains all asynchronous bridges for crossing clock domains. The CPU bridge is split, with one half of each bridge in the core clock domain and the other half in the relevant cluster domain. All core I/O signals go through the CPU bridge and the SYS power domain.

Clamping cells between power domains are inferred through power intent files rather than instantiated in the RTL.

The following figure shows the organization of the power domains for a single Cortex®-A65AE core pair. The colored boxes indicate the PDADVSIMD, PDCPU, and PDSYS power domains, with respective voltage domains shown in dotted lines.

**Figure 2-6: Cortex®-A65AE core power domains in Split-mode**



For Lock-mode, the cores in this figure would be named Core 0 and Core 0'.

See the *Arm® Cortex®-A65AE Core Configuration and Sign-off Guide* for more information on power domains.

## 2.4.4 Architectural clock gating modes

When the core is in Standby mode, it is architecturally clock gated at the top of the clock tree.

*Wait for Interrupt* (WFI) and *Wait for Event* (WFE) are features of the Arm®v8-A architecture that put the core in a low-power Standby mode by architecturally disabling the clock at the top of the clock tree.

If both threads on the core are in WFI or WFE mode, then all registers in the core do not receive a clock signal.

Each thread has a power mode, which is either the Run, Standby, or Deactivated thread power mode. For more information, see [2.4.7 Thread power modes](#) on page 44.

If one of the threads on the core executes a `WFI` or `WFE` instruction, and there is no pending wakeup event for that thread, the following occurs:

- All preceding instructions for the thread are completed and retired.
- All following instructions for the thread are flushed.
- The thread moves into Standby mode.

The core begins to quiesce when either of the following occurs:

- Both of the threads are in Standby mode.
- One thread is in Standby mode, and the other thread is in Deactivated mode.

The core is put into the lowest power state legally possible:

- If both threads are in WFI or WFE, then the core is put into WFI or WFE respectively.
- If one thread is in WFE and the other is in WFI, then the core is put into WFE.
- If one thread is in Deactivated mode, and the other thread is in WFI or WFE, then the core is put into WFI or WFE respectively.

### 2.4.4.1 Core Wait for Interrupt

If the core is in WFI, it moves into a low-power state by disabling most of the clocks in the core, while keeping the core powered up.

There is a small dynamic power overhead from the logic that is required to wake up the core from WFI low-power state. Other than this, the power that is drawn is reduced to static leakage current only.

When a thread on the core executes the `WFI` instruction, the thread waits for all instructions in the core to retire before it enters Standby mode. When both threads are in Standby mode, or one is in Standby mode and the other is off, the core will begin to quiesce. This process ensures that all explicit memory accesses that occurred before the `WFI` instruction have retired in program order. In addition, this process ensures that store instructions have updated the cache or have been issued to the L3 memory system.

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

- An L3 snoop request that must be serviced by the core data caches.
- A cache or TLB maintenance operation that must be serviced by the core L1 instruction cache, data cache, TLB, or L2 cache.
- An APB access to the debug or trace registers residing in the core power domain.
- A GIC CPU access through the AXI4 stream channel.

Exit from WFI low-power state occurs when one of the following occurs:

- The core detects one of the WFI wake-up events.
- The core detects a reset.

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

#### 2.4.4.2 Core Wait for Event

WFE is a feature of the Arm®v8-A architecture. It uses a locking mechanism based on events, to put the core in a low-power state by disabling most of the clocks in the core, while keeping the core powered up.

There is a small dynamic power overhead from the logic that is required to wake up the core from WFE low-power state. Other than this, the power that is drawn is reduced to static leakage current only.

A thread on the core enters into WFE low-power state by executing the `WFE` instruction. When the `WFE` instruction executes, the thread waits for all instructions in the core to complete before it enters the Standby mode.

If the event register is set, execution of WFE does not cause entry into Standby mode, but clears the event register. When both threads are in Standby mode, or one is in Standby mode and the other is off, the core will begin to transition to WFE low-power state.

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

- An external snoop request that must be serviced by the core data caches.
- A cache or TLB maintenance operation that must be serviced by the core L1 instruction cache, data cache, TLB, or L2 cache.
- An APB access to the debug or trace registers residing in the core power domain.
- A GIC CPU access through the AXI4 stream channel.

Exit from WFE low-power state occurs when one of the following occurs:

- The core detects one of the WFE wake-up events.
- The **EVENTI** input signal is asserted.

- The core detects a reset.

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

## Related information

[Power down sequence](#) on page 46

### 2.4.5 Power control

All power mode transitions are performed at the request of the power controller, using a P-Channel interface to communicate with the Cortex®-A65AE core.

There is one P-Channel per core, plus one P-Channel for the cluster. The Cortex®-A65AE core provides the current requirements on the **PACTIVE** signals, so that the power controller can make decisions and request any change with **PREQ** and **PSTATE**. The Cortex®-A65AE core then performs any actions necessary to reach the requested power mode, such as gating clocks, flushing caches, or disabling coherency, before accepting the request.

If the request is not valid, either because of an incorrect transition or because the status has changed so that state is no longer appropriate, then the request is denied. The power mode of each core can be independent of other cores in the cluster, however the cluster power mode is linked to the mode of the cores.

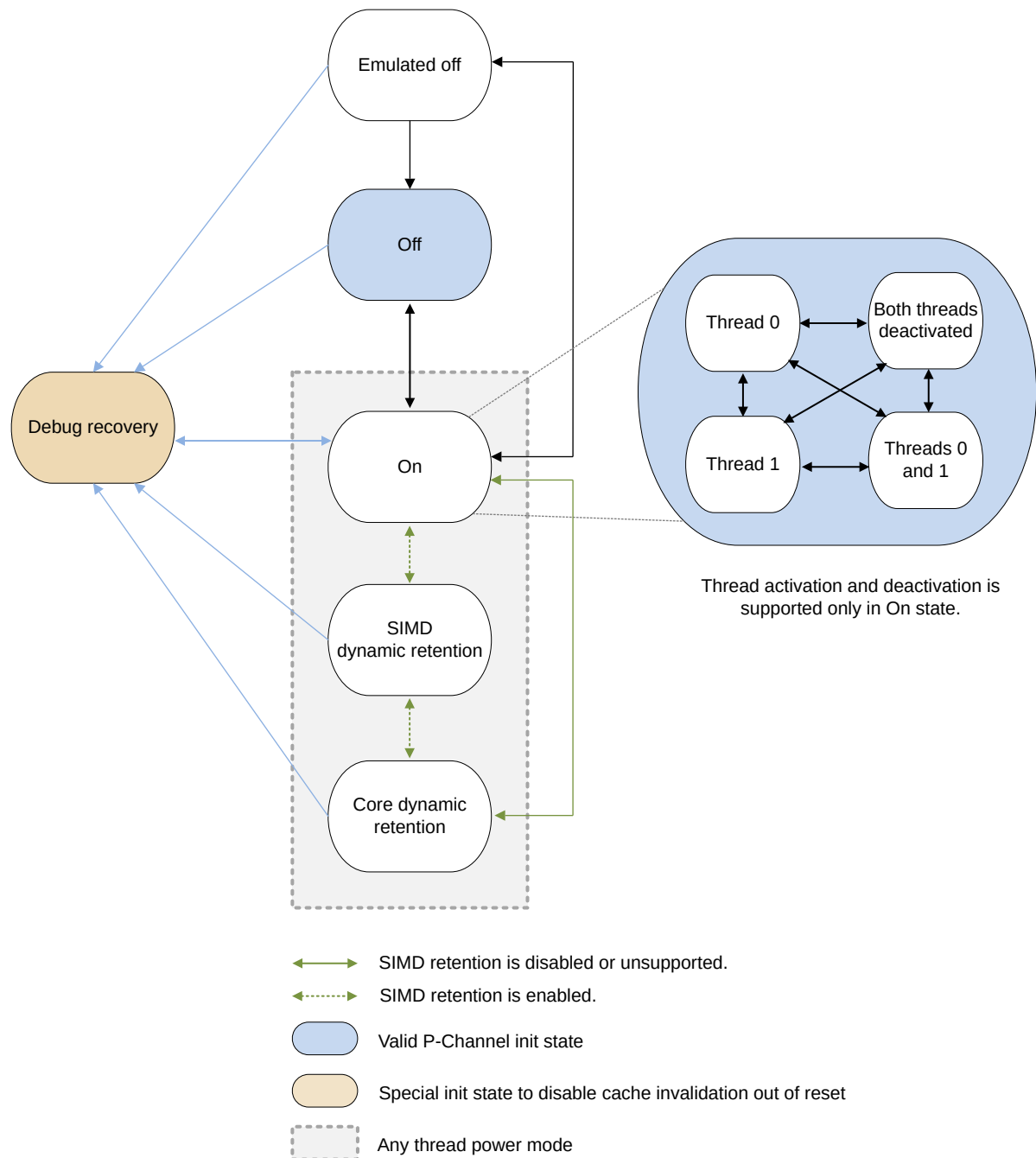
### 2.4.6 Core power modes

The Cortex®-A65AE core supports the following core power modes for each core domain P-Channel:

- [2.4.6.1 On mode](#) on page 39.
- [2.4.6.2 Off mode](#) on page 40.
- [2.4.6.3 Emulated off mode](#) on page 40.
- [2.4.6.4 SIMD dynamic retention mode](#) on page 40.
- [2.4.6.5 Core dynamic retention mode](#) on page 41.
- [2.4.6.6 Debug recovery mode](#) on page 42.
- [2.4.6.7 Encoding for power modes](#) on page 42.

There are legal transitions between each core power mode. The core can only be in certain core power modes, depending on which thread power modes (run, standby, or deactivated mode) both threads are in. For more information on the corresponding thread power modes for each core power mode and power domain, see [2.4.8 Relationship between power modes and power domains](#) on page 45.

The following figure shows the supported legal transitions between the core power modes, and the corresponding thread power modes that the threads should be in.

**Figure 2-7: Cortex®-A65AE core power mode transitions**

For information on core power mode encodings, see [2.4.6.7 Encoding for power modes](#) on page 42.

### 2.4.6.1 On mode

In this mode, the core is on and fully operational. Each thread can be in Run, Standby, or Deactivated mode.

The core can be initialized into the On mode with the required threads enabled. If the core does not use its P-Channel, you can tie the core in the On mode by tying **PREQ** LOW.

When a transition to the On mode completes, all caches are accessible and coherent. Other than the normal architectural steps to enable caches, no additional software configuration is required.

When the core domain P-Channel is initialized into the On mode, either as a shortcut for entering that mode or as a tie-off for an unused P-Channel, it is an assumed transition from the Off mode. This includes an invalidation of any cache RAM within the core domain.

### 2.4.6.2 Off mode

The Cortex®-A65AE core supports a full Shutdown mode where power can be removed completely and no state is retained.

The shutdown can be for either the whole cluster, for an individual core when in Split-mode, or for a core pair when in Lock-mode.

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

A Cold reset can reset the core in this mode.

The core P-Channel can be initialized into this mode.

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

### 2.4.6.3 Emulated off mode

In this mode, both threads are deactivated, and core domain logic and RAMs are kept on. However, a core Warm reset can be asserted externally to emulate a powerdown scenario while keeping core debug state and allowing debug access.

All Debug registers must retain their mode and be accessible from the external debug interface. All other functional interfaces behave as if the core was in Off mode.



#### 2.4.6.4 SIMD dynamic retention mode

In this mode, the Advanced SIMD and floating-point logic is in retention (inoperable but with state retained) and the remainder of the core logic is operational.

This means that if an Advanced SIMD and floating-point instruction in either thread is executed while in this mode, it is stalled until the core enters the On mode.

When the Advanced SIMD and floating-point logic is in retention, the clock to the logic is automatically gated outside of the retained domain.

The SIMD dynamic retention mode is controlled by the CPUPWRCTLR.SIMD\_RET\_CTRL bit.

#### Related information

CPUPWRCTLR\_EL1, [Power Control Register, EL1](#) on page 131

#### 2.4.6.5 Core dynamic retention mode

In this mode, all core logic and RAMs are in retention and the core domain is inoperable. The core can enter this power mode when both threads are in WFI mode, both threads are in WFE mode, or when one thread is in WFI mode and the other thread is in WFE mode.

For both Split-mode and Lock-mode, Core dynamic retention mode must be enabled at the core pair level granularity.

The Core dynamic retention mode can be enabled and disabled separately for WFI and WFE by software running on each thread in the core. Separate timeout values can be programmed for entry into this mode from WFI and WFE mode:

- Use the CPUPWRCTLR.WFI\_RET\_CTRL register bits to program timeout values for entry into Core dynamic retention mode from WFI mode.
- Use the CPUPWRCTLR.WFE\_RET\_CTRL register bits to program timeout values for entry into Core dynamic retention mode from WFE mode.

The clock to the core is automatically gated outside of the domain when the core is in Core dynamic retention mode and is running synchronously to the cluster. However, if the core is running asynchronously to the cluster, the system integrator must gate the clock externally during Core dynamic retention mode. For more information, see the *Arm® DynamIQ™ Shared Unit AE Configuration and Sign-off Guide*.

The outputs of the domain must be isolated to prevent buffers without power from propagating **UNKNOWN** values to any operational parts of the system.

When the core is in Core dynamic retention mode there is support for snoop, GIC, and debug access, so the core appears as if it were in WFI or WFE mode. When an incoming access occurs, it stalls, and the On **PACTIVE** bit is set HIGH. The incoming access proceeds when the domain is returned to the On mode using P-Channel.

When the incoming access completes, and if the core has not exited WFI or WFE mode, then the On **PACTIVE** bit is set LOW after the programmed retention timeout. The power controller can then request to reenter the Core dynamic retention mode.

### Related information

[CPUPWRCTLR\\_EL1, Power Control Register, EL1](#) on page 131

#### 2.4.6.6 Debug recovery mode

Debug recovery mode can be used to assist debug of external watchdog-triggered reset events.

It allows contents of the core L1 instruction, L1 data and L2 caches 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.

By default, the core invalidates its caches when transitioning from Off to On mode. If P-Channel is initialized to Debug recovery mode, and the core is cycled through Cold or Warm reset along with system resets, then the cache invalidation is disabled. The cache contents are preserved when the core is transitioned to the On mode.

Debug recovery mode also supports preserving *Reliability, Availability, and Serviceability* (RAS) state, in addition to the cache contents. In this case, a transition to Debug recovery mode is made from any of the current states. Once in Debug recovery mode, a cluster-wide Warm reset must be applied externally. The RAS and cache state are preserved when the core is transitioned to the On mode.

---

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



Caution

- This mode can occur at any time with no guarantee of the state of the core. A P-Channel request of this type is accepted immediately, therefore its effects on the core, cluster, or the wider system are unpredictable, and a wider system reset might be required. In particular, if there were outstanding memory system transactions at the time of the reset, then these might complete after the reset when the core is not expecting them and cause a system deadlock.
  - If the system sends a snoop to the cluster during this mode, then depending on the cluster state, the snoop might get a response and disturb the contents of the caches, or it might not get a response and cause a system deadlock.
-

### 2.4.6.7 Encoding for core power modes

The following table shows the encodings for the supported modes for each core domain P-Channel.

**Table 2-4: Core power modes COREPSTATE encoding**

Core power mode name	Short Name	PACTIVE Bit Number	PSTATE[5:4] value	PSTATE[3:0] value	Description
Debug recovery	DBG_RECOV	-	0b00	0b1010	Threads must be deactivated, and PSTATE[5:4] must be 0b00.  Logic is off (or in reset), RAM state is retained and not invalidated when transition to On mode.
On	ON	8	0b00 0b01 0b10 0b11	0b1000	All powerup, each thread can be in Run, Standby, or Deactivated mode.  PSTATE[5:4]:  0b00 Both threads are deactivated. 0b01 Thread 0 is activated. 0b10 Thread 1 is activated. 0b11 Both threads are activated.
SIMD dynamic retention	SIMD_DYN_RET	7	0b01 0b10 0b11	0b0111	SIMD logic is in retention and inoperable. All other logic is on and operational.  PSTATE[5:4]:  0b01 Thread 0 is activated. 0b10 Thread 1 is activated. 0b11 Both threads are activated.
Core dynamic retention	CORE_DYN_RET	5	0b01 0b10 0b11	0b0101	Logic and RAM state are inoperable but retained.  PSTATE[5:4]:  0b01 Thread 0 is activated. 0b10 Thread 1 is activated. 0b11 Both threads are activated.
Emulated off	EMU_OFF	1	0b00	0b0001	Threads must be deactivated, and PSTATE[5:4] must be 0b00.  On with Warm reset asserted, debug state is retained and accessible.
Off	OFF	0 (implicit)	0b00	0b0000	Threads must be deactivated, and PSTATE[5:4] must be 0b00.  All powerdown.



As part of the powerdown sequence any core in lock-step in the A65AE must disable and clear any interrupt outputs from the core, such as the timer interrupts. Failing to clear these interrupt outputs can lead to a false positive error report from the lock-step comparators.

## 2.4.7 Thread power modes

Each thread on a Cortex®-A65AE core has a power mode. The combination of these modes drive the overall core power mode.

### 2.4.7.1 Run mode

Run mode is the normal mode of thread operations.

In Run mode, the thread is active and executes instructions.

### 2.4.7.2 Standby mode

A thread enters Standby mode through the execution of a `WFI` or `WFE` instruction.

In Standby mode, the thread and its architectural state are active. However, the instruction execution is suspended.

A wake-up event causes the thread to reenter Run mode.

### 2.4.7.3 Deactivated mode

The Cortex®-A65AE core uses its P-Channel to activate or deactivate threads.

If a thread powers down by setting the `CPUPWRCTLR.CORE_PWRDN_EN` bit HIGH and executing `WFI`, then the power controller can deactivate the thread.

The power controller uses **PSTATE[5:4]** bits to activate or deactivate both or any of the two threads in the same P-Channel request.

**PACTIVE[17:16]** bits indicate to the power controller whether the threads are required to be active. If a thread executes the deactivated sequence, then the corresponding **PACTIVE** bit for that thread is set LOW. The power controller then sets the matching **PSTATE** bit LOW. This arrangement allows a system power controller to treat each thread as a separate core and manage powerup or powerdown requests for each thread.

If one of the threads executes a deactivated sequence, the core enters a single-threaded linked mode. In this mode, the core can allocate all resources to the remaining thread, which achieves higher performance.

In deactivated mode, the thread is inactive, and all architectural states are **UNKNOWN**.

The following table shows the corresponding **PSTATE** and **PACTIVE** bits for respective threads.

**Table 2-5: PSTATE and PACTIVE bits for corresponding threads**

PSTATE[5]	PSTATE[4]	PACTIVE[17]	PACTIVE[16]	Description
0b0	0b0	LOW	LOW	Thread 0 and 1 are deactivated.
0b0	0b1	LOW	HIGH	Thread 0 is activated. The core enters a single-threaded linked mode.
0b1	0b0	HIGH	LOW	Thread 1 is activated. The core enters a single-threaded linked mode.
0b1	0b1	HIGH	HIGH	Thread 0 and 1 are activated.

Threads are active only when the core power modes are in any of the following states:

- On.
- SIMD dynamic retention.

A thread can be activated by a P-Channel request with the corresponding **PSTATE[5:4]** bits set HIGH while the core power mode is On.

A thread can be deactivated by a P-Channel request with the corresponding **PSTATE[5:4]** bits set LOW while the core power mode is On.

## 2.4.8 Relationship between power modes and power domains

The following table describes the core power modes, and the corresponding thread power modes and power domain states for individual cores.

The power domains can be controlled independently to give different combinations when powered-up and powered-down. However, only some powered-up and powered-down domain combinations are valid and supported.



Caution

States that are not shown in the following table are unsupported and must not occur.

**Table 2-6: Supported combinations of core power mode, thread power mode, and power domain states**

Core power mode	Thread power mode <sup>1</sup>	Power domain		Description
		PDCPU	PDADVSIMD	
Debug recovery	Deactivated for both threads	On	On	Core on. Advanced SIMD and floating-point logic on. Logic is active.
On	Run, standby, or deactivated	On	On	Core on. Advanced SIMD and floating-point logic on. Logic is active. Thread power modes are independent.
SIMD dynamic retention	Run, standby, or deactivated	On	Ret	Core on. Advanced SIMD and floating-point logic in retention. Logic is active.

<sup>1</sup> For more information on the relationship between the core power modes and thread power modes, see [2.4.6.7 Encoding for power modes](#) on page 42 and [2.4.7 Thread power modes](#) on page 44

Core power mode	Thread power mode <sup>1</sup>	Power domain		Description
		PD CPU	PD ADVSIMD	
Core dynamic retention	Deactivated for both threads	Ret	Ret	Core retention. Core logic and Advanced SIMD and floating-point logic in retention. Logic and RAM retention power only.
Emulated off	Deactivated for both threads	On	On	Core on. Advanced SIMD and floating-point logic on. Logic is inactive.
Off	Deactivated for both threads	Off	Off	Core off. Power to the logic is gated.

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

## 2.4.9 Power down sequence

The Cortex®-A65AE core uses the following power down sequence.

To power down a core, perform the following programming sequence on each thread:

1. Save all architectural state.
2. To disable or reroute interrupts away from this thread, configure the GIC distributor.
3. To indicate to the power controller that a powerdown is requested, set the CPUPWRCTLR.CORE\_PWRDN\_EN bit to 1.
4. Execute an *Instruction Synchronization Barrier* (ISB) instruction.
5. Execute a WFI instruction.

After a WFI executes, hardware, under the direction of the power controller, perform all cache disables, cache flushing (L1 and L2), and removal of the core from coherency.



Note

When the CPUPWRCTLR.CORE\_PWRDN\_EN bit is set and WFI instruction is executed, the thread can be deactivated under the direction of power controller. This masks all interrupts and wake-up events in the core for that thread. The power controller can activate the thread via a P-Channel thread activation request.

If one of the threads executes a power down sequence, the core enters a single-threaded mode. In this mode, the core can allocate all resources to the remaining thread which achieves higher performance.

### Related information

[CPUPWRCTLR\\_EL1, Power Control Register, EL1](#) on page 131

<sup>1</sup> For more information on the relationship between the core power modes and thread power modes, see [2.4.6.7 Encoding for power modes](#) on page 42 and [2.4.7 Thread power modes](#) on page 44

## 2.4.10 Debug over powerdown

The Cortex®-A65AE core supports debug over powerdown, which allows a debugger to retain its connection with the core even when powered down. This 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, which is external to the cluster and can be implemented in a separate power domain. If the DebugBlock is in the same power domain as the core, then debug over powerdown is not supported.

For more information on the DebugBlock, see the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual*.



In Lock-mode, debug over powerdown might generate a *Dual-Core Lock-Step* (DCLS) mismatch if debug activity is in progress during Warm reset, that is, during **nCORERESET** assertion.

## 2.5 Memory Management Unit

This chapter describes the *Memory Management Unit* (MMU) of the Cortex®-A65AE core.

### 2.5.1 About the MMU

The *Memory Management Unit* (MMU) is responsible for translating addresses of code and data *Virtual Addresses* (VAs) to *Physical Addresses* (PAs) in the real system. The MMU also controls memory access permissions, memory ordering, and cache policies for each region of memory.

#### 2.5.1.1 Main functions

The three main functions of the MMU are to:

- Control the translation table walk hardware that accesses translation tables in main memory.
- Translate *Virtual Addresses* (VAs) to *Physical Addresses* (PAs).
- Provide fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes that are held in translation tables.

Each stage of address translation uses a set of address translations and associated memory properties that are held in memory mapped tables called translation tables. Translation table entries can be cached into a *Translation Lookaside Buffer* (TLB).

The following table describes the components included in the MMU.

**Table 2-7: TLBs and TLB caches in the MMU**

Component	Description
Instruction L1 TLB	20 entries, fully associative
Data L1 TLB	32 entries, fully associative
L2 TLB	1024 entries, 4-way set associative
Walk cache RAM	64 entries, 4-way set associative
IPA cache RAM	64 entries, 4-way set associative

L2 TLB entries contain global and *Address Space Identifiers* (ASID) to prevent context switch TLB flushes.

The TLB entries contain a *Virtual Machine Identifier* (VMID) to prevent context switch TLB flushes on virtual machine switches by the hypervisor.

The Cortex®-A65AE core supports a 44-bit physical address range, which allows 16TB of physical memory to be addressed.

### 2.5.1.2 AArch64 MMU behavior

The Cortex®-A65AE core is an Arm®v8-A compliant core that supports execution in AArch64 state only.

The following table shows the behavior in AArch64 state.

**Table 2-8: AArch64 MMU behavior**

	AArch64
Address translation system	The Arm®v8-A address translation system resembles an extension to the Long descriptor format address translation system to support the expanded virtual and physical address space.
Translation granule	4KB, 16KB, or 64KB
ASID size	8 or 16 bits, depending on the value of TCR_ELx.AS.
VMID size	8 or 16 bits, depending on the value of VTCR_EL2.VS.
PA size	Maximum 44 bits.

The Cortex®-A65AE core also supports the *Virtualization Host Extension* (VHE) including ASID space for EL2. When VHE is implemented and enabled, EL2 has the same behavior as EL1.

See the *Arm® Architecture Reference Manual Arm®v8, for Arm®v8-A architecture profile* for more information on concatenated translation tables and for address translation formats.

### 2.5.2 TLB organization

The *Translation Lookaside Buffer* (TLB) is a cache of recently executed page translations within the MMU. The Cortex®-A65AE core implements a two-level TLB structure. The L2 TLB stores all page



sizes and is responsible for breaking these down into smaller pages when required for the data-side or instruction-side L1 TLB.

TLB lockdown is not supported.

After reset, an Invalidate All operation is executed and all entries in the TLB are invalidated.

### 2.5.2.1 L1 TLB

The first level of caching for the translation table information is an L1 TLB, implemented on each of the instruction and data sides.

The Cortex®-A65AE L1 instruction TLB supports 4KB, 16KB, 64KB, and 2MB pages.

The Cortex®-A65AE L1 data TLB supports 4KB, 16KB, and 64KB pages.

Any other page sizes are fractured after the L2 TLB and the appropriate page size sent to the L1 TLB.

All TLB maintenance operations affect both the L1 instruction and data TLBs and cause them to be invalidated.

### 2.5.2.2 L2 TLB

A unified L2 TLB handles any misses from the L1 instruction and data TLBs.

- A 4-way, set-associative, 1024-entry cache.
- Supports all *Virtual Memory System Architecture* (VMSA) block sizes as described in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*. See VMSAv8-64 in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information.

Accesses to the L2 TLB take a variable number of cycles, based on:

- Competing requests from the L1 TLBs.
- TLB maintenance operations in flight.
- Different page size mappings in use.

### 2.5.2.3 IPA cache RAM

The *Intermediate Physical Address* (IPA) cache RAM holds mappings between the IPAs and *Physical Addresses* (PAs).

Only Non-secure EL1 and EL0 stage 2 translations use the IPA cache. When a stage 2 translation completes, the cache is updated. The IPA cache is checked whenever a stage 2 translation is required.

Like the L2 TLB, the IPA cache RAM can hold entries for different sizes.

#### 2.5.2.4 Walk cache RAM

The walk cache RAM holds the result of a stage 1 translation up to, but not including, the last level.

### 2.5.3 TLB match process

The Arm®v8-A architecture provides support for multiple maps from the VA space that are translated differently.

TLB entries store the context information that is required to facilitate a match and avoid the need for a TLB flush on a context or virtual machine switch.

Each TLB entry contains a:

- VA.
- PA.
- Set of memory properties that include type and access permissions.

Each entry is either associated with a particular *Address Space Identifier* (ASID) or is global. In addition, each TLB entry contains a field to store the *Virtual Machine Identifier* (VMID) in the entry applicable to accesses from Non-secure EL0 and EL1 Exception levels.

Each entry is associated with a particular translation regime.

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

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

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

- The ASID and VMID matches are **IGNORED** when ASID and VMID are not relevant. ASID is relevant when the translation regime is:

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

VMID is relevant for EL1 or EL0 in Non-secure state when HCR\_EL2.E2H and HCR\_EL2.TGE are not both set.

## 2.5.4 Translation table walks

When an access is requested at an address, 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.

When the Cortex®-A65AE core generates a memory access, the following process occurs:

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

In the case of an L2 TLB miss, the hardware does a translation table walk as long as the MMU is enabled, and the translation using the base register has not been disabled.

If the translation table walk is disabled for a particular base register, the core returns a Translation Fault. If the TLB finds a matching entry, it uses the information in the entry as follows.

The access permission bits and the domain determine if the access is permitted. If the matching entry does not pass the permission checks, the MMU signals a Permission fault. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for details of Permission faults, including:

- A description of the various faults.
- The fault codes.
- Information regarding the registers where the fault codes are set.

### Page granule

When executing at a particular Exception level, you can configure the hardware translation table walk to use either the 4KB, 16KB, or 64KB translation granule. Program the Translation Granule bit, TGO, in the appropriate translation control register:

- TCR\_EL1.

- TCR\_EL2.
- TCR\_EL3.
- VTCR\_EL2.

For TCR\_EL1, you can program the Translation Granule bits TGO and TG1 to configure the translation granule respectively for TTBR0\_EL1 and TTBR1\_EL1, or TCR\_EL2 when VHE is enabled.

## 2.5.5 MMU memory accesses

During a translation table walk, the *Memory Management Unit* (MMU) generates memory accesses. The Cortex®-A65AE core has specific behaviors for MMU memory accesses.

### 2.5.5.1 Configuring MMU accesses

By programming the IRGN and ORGN bits, you can configure the MMU in the appropriate TCR\_ELx register to perform translation table walks in cacheable or Non-cacheable regions:

If the encoding of both the ORGN and IRGN bits is Write-Back, the data cache lookup is performed and data is read from the data cache. External memory is accessed, if the ORGN and IRGN bit contain different attributes, or if the encoding of the ORGN and IRGN bits is Write-Through or Non-cacheable.

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

The Cortex®-A65AE core includes the option to perform hardware updates to the translation tables.

These features are enabled in registers TCR\_ELx and VTCR\_EL2. To support the hardware management of dirty state, the *Dirty Bit Modifier* (DBM) field is added to the translation table descriptors as part of Arm®v8.1-A architecture.

The core supports hardware updates to the Access flag and to dirty state only when the translation tables are held in Inner Write-Back, 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 core returns an abort with the following encoding:

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

The following table shows the conditions that can cause hardware updates to the Access flag or dirty state.

**Table 2-9: Conditions that can cause hardware updates to the Access flag or dirty state**

Conditions				Hardware update in the translation table
Instruction to the memory location	DBM	Access permission	Situation	
Store-Exclusive	DBM bit is 0b1	Stage 1 AP[2] bit is 0b1	The Store-Exclusive fails because the exclusive monitor is not in the exclusive state.	AP[2] bit is updated.
		Stage 2 S2AP[1] bit is 0b0		S2AP[1] bit is updated.
Store	DBM bit is 0b1	Stage 1 AP[2] bit is 0b1	The memory location generates any of the following: <ul style="list-style-type: none"> <li>A synchronous external abort on a write for a store to a memory location.</li> <li>A watchpoint on a write.</li> </ul>	AP[2] bit is updated.
		Stage 2 S2AP[1] bit is 0b0		S2AP[1] bit is updated.
Compare and Swap (CAS or CASP)	DBM bit is 0b1	Stage 1 AP[2] bit is 0b1	The compare fails and the location is not updated.	AP[2] bit is updated.
		Stage 2 S2AP[1] bit is 0b0		S2AP[1] bit is updated.

For more information about hardware updates of the Access flag and dirty state, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 2.5.6 Responses

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

### 2.5.6.1 MMU responses

When one of the following translations is completed, the MMU generates a response to the requester:

- An L1 TLB hit.
- An L2 TLB hit.
- A translation table walk.

The response from the MMU contains the following information:

- The PA corresponding to the translation.
- A set of permissions.
- Secure or Non-secure.
- All the information required to report aborts. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more details.

### 2.5.6.2 MMU aborts

The MMU can detect faults that are related to address translation and can cause exceptions to be taken to the processing element. Faults can include address size, translation, access flags, and permissions.

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

### 2.5.6.3 External aborts

External aborts are aborts that occur in the memory system rather than aborts that the MMU detects. Normally, external memory aborts are rare. External aborts are caused by errors flagged by the external memory interfaces or are generated because of an uncorrected ECC error in the L1 data cache or L2 cache arrays.

When an external abort to the external interface occurs on an access for a translation table walk access, the MMU returns a synchronous external abort. For a Load Multiple or a Store Multiple operation, the address captured in the fault register is that of the address that generated the synchronous external abort.

### 2.5.6.4 Mis-programming contiguous hints

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

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

If there is this kind of mis-programming, the Cortex®-A65AE core does not generate a translation fault.

### 2.5.6.5 Conflict aborts

If a conflict abort is detected in the L2 TLB, the MMU chooses one valid translation and does not generate a conflict abort.

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

### 2.5.6.6 Memory Behavior

The Cortex®-A65AE core supports all the Arm®v8-A memory types.

However, the following behaviors are simplified and so for best performance their use is not recommended:

**Write-Through** Memory that is marked as Write-Through cannot be cached on the data-side and does not make coherency requests. On the instruction-side, areas that are marked as Write-Through and Write-Back can be cached in the L1 instruction cache. However, only areas marked as Write-Back can be cached in the L2 cache or the L3 cache.

**Mixed inner and outer** Memory that is not marked as inner and outer Write-Back cannot be cached on the data-side and does not make coherency requests. This applies to the memory cacheability type only, and not to the allocation hints. All caches within the cluster are treated as being part of the inner cacheability domain.

For more information on supported memory behaviors, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*

### 2.5.6.7 Support for Arm®v8-A device memory types

The Arm®v8-A architecture includes memory types that replace the Arm®v7 Device and Strongly-ordered memory types. These device memory types have the following three 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 acknowledge of transactions from the interconnect.

The legal combinations are described in the following table:

**Table 2-10: Arm®v8-A Device Memory Types**

Memory type	Cortex®-A65AE support	Comment
GRE	Yes	Similar to Normal non-cacheable, but does not permit speculative accesses.
nGRE	Yes	Transactions might be reordered within the L3 memory system, or in the system interconnect.
nGnRE	Yes	Corresponds to Device in Arm®v7.
nGnRnE	Yes	Corresponds to Strongly Ordered in Arm®v7.  Treated the same as nGnRE inside the Cortex®-A65AE core, but reported differently on the bus interface.

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

## 2.6 L1 memory system

This chapter describes the L1 instruction cache and data cache that make up the L1 memory system.

### 2.6.1 About the L1 memory system

The L1 memory system enhances the performance and power efficiency in the Cortex®-A65AE core.

It consists of separate instruction and data caches. You can configure instruction and data caches independently during implementation to sizes of 32KB or 64KB.

#### L1 instruction-side memory system

The L1 instruction-side memory system provides an instruction stream to the DPU. Its key features are:

- 64-byte instruction side cache line length.
- 4-way set associative L1 instruction cache.
- 128-bit read interface to the L2 memory system.

The Cortex®-A65AE core uses extensive branch prediction to improve *Instructions Per Clock* (IPC) and power efficiency.

#### L1 data-side memory system

The L1 data-side memory system responds to load and store requests from the DPU. It also responds to SCU snoop requests from other cores, or external masters. Its key features are:

- 64-byte data side cache line length.
- 4-way set associative L1 data cache.
- Read buffer that services both the *Data Cache Unit* (DCU), and the *Instruction Fetch Unit* (IFU).
- 128-bit read path from the data L1 memory system to the datapath.
- 512-bit write path from the datapath to the L1 memory system.
- Merging store buffer capability which writes to all types of memory (device, normal cacheable and normal non-cacheable).
- Data side prefetch engine that detects patterns of strides with multiple streams are allowed in parallel, capable of detecting both constant and patterns of strides.



## 2.6.2 Cache behavior

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

This word-first fill is an implementation-specific feature of the instruction and data caches.

### 2.6.2.1 Instruction cache disabled behavior

If the instruction cache is disabled, all instruction fetches to cacheable memory are treated as if they were non-cacheable.

This means that instruction fetches might not be coherent with caches in other cores, and software must take account of this.

Lines may still be allocated into the instruction cache even if the memory is marked non-cacheable or the instruction cache is disabled. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information.

### 2.6.2.2 Instruction cache speculative memory accesses

Instruction fetches are speculative. Execution is not guaranteed, because there can be several unresolved branches in the pipeline.

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

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

If the instruction cache is enabled, and if the instruction fetches miss in the L1 instruction cache, they can still look up in the L1 data caches. However, a new line is not allocated in the data cache unless the data cache is enabled.

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

### 2.6.2.3 Data cache disabled behavior

If the SCTL.R.C bit is set to 0, load and store instructions do not access any of the L1 data, L2, or DSU-AE L3 caches.

The SCTL.R.C bit controls whether accesses from the core can look up and allocate into the data cache and unified L2 or L3 caches. Data cache maintenance operations execute normally, regardless of how the SCTL.R.C bit is set.

If the SCTLR.C bit is set to 0, then the following apply:

- Instruction fetches cannot allocate in the L2 or L3 caches.
- All load and store instructions to cacheable memory are treated as if they were non-cacheable. Therefore, they are not coherent with the caches in this core or the caches in other cores, and software must take this into account.

The L2 and L1 data caches cannot be disabled independently.

For more information, see [Other system control registers](#) on page 98.

### 2.6.2.4 Data cache maintenance considerations

DC IVAC instructions perform an invalidate of the target address.

If the data is dirty within the cluster, a clean is performed before the invalidate.

DC ISW and DC CSW instructions perform both a clean and invalidate of the target set/way. The values of HCR.SWIO and HCR\_EL2.SWIO have no effect.

### 2.6.2.5 Data cache coherency

The Cortex®-A65AE core uses the MESI protocol to maintain data coherency between multiple cores.

MESI describes the state that a shareable line in a L1 data cache can be in:

M	Modified/ <i>UniqueDirty</i> (UD). The line is in only this cache and is dirty.
E	Exclusive/ <i>UniqueClean</i> (UC). The line is in only this cache and is clean.
S	Shared/ <i>SharedClean</i> (SC). The line is possibly in more than one cache and is clean.
I	Invalid/ <i>Invalid</i> (I). The line is not in this cache.

The DCU stores the MESI state of the cache line in the tag and dirty RAMs.



The names *UniqueDirty*, *SharedDirty*, *UniqueClean*, *SharedClean*, and *Invalid* are the AMBA names for the cache states. The Cortex®-A65AE core does not use the *SharedDirty* AMBA state.

---

### 2.6.2.6 Write Streaming Mode

A cache line is allocated to the L1 on either a read miss or a write miss.

However, there are some situations where allocating 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. Writes of large blocks of data can pollute the cache with unnecessary data. It can also waste

power and performance if a linefill must be performed only to discard the linefill data because the entire line was subsequently written by the `memset()`.

To counter this, the BIU includes logic to 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. This is sometimes referred to as read allocate mode.

When in write streaming mode, loads will behave as normal, and can still cause linefills, and writes will still lookup in the cache, but if they miss then they will write out to L2 (or possibly L3) rather than starting a linefill.



More than the specified number of linefills might be observed on the ACE or CHI master interface, before the BIU detects that three full cache lines have been written and switches to write streaming mode.

---

The BIU continues in write streaming mode until it detects either a cacheable write burst that is not a full cache line, or there is a load from the same line as is currently being written to L2.

When a core has dropped into write streaming mode, the BIU continues to monitor the bus traffic and will signal to the L2 for it to go into write streaming mode when a further number of full cache line writes are seen.

The Cortex®-A65AE core is a multithreaded core. Entering write streaming mode is managed independently by each of the two threads.

#### **AArch64 state**

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

For more information, see [3.1.25 CPUECTLR\\_EL1, CPU Extended Control Register, EL1](#) on page 121.

#### **2.6.2.7 Data cache invalidate on reset**

The Arm®v8-A architecture does not support an operation to invalidate the entire data cache.

The Cortex®-A65AE core automatically invalidates caches on reset unless suppressed with the debug recovery P-Channel state. It is therefore not necessary for software to invalidate the caches on startup.

## 2.6.3 L1 instruction memory system

The L1 instruction side memory system provides an instruction stream to the *Data Processing Unit* (DPU).

To increase overall performance and to reduce power consumption, it uses:

- Dynamic branch prediction.
- Instruction caching.

### 2.6.3.1 Program flow prediction

The Cortex®-A65AE core contains program flow prediction hardware, also known as branch prediction.

Branch prediction increases overall performance and reduces power consumption. With program flow prediction disabled, all taken branches incur a penalty that is associated with flushing the pipeline.

To avoid this penalty, the branch prediction hardware predicts if a conditional or unconditional branch is to be taken. For conditional branches, the hardware predicts if the branch is to be taken. It also predicts the address that the branch goes to, known as the branch target address. For unconditional branches, only the target is predicted.

The hardware contains the following functionality:

- A BTAC holding the branch target address of previously taken branches.
- Dynamic branch predictor history.
- The return stack, a stack of nested subroutine return addresses.
- A static branch predictor.
- An indirect branch predictor.

#### Predicted and non-predicted instructions

Unless otherwise specified, the following list applies to A64 instructions. As a rule the flow prediction hardware predicts all branch instructions regardless of the addressing mode, and includes:

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

The following branch instructions are not predicted:

- Exception return instructions.

#### Return stack

The return stack stores the address and instruction set state.

This address is equal to the link register value stored in X30.

In AArch64 state, the `RET` instruction causes a return stack pop.

As exception return instructions can change core privilege mode and Security state, they are not predicted. These include:

- `ERET`

## 2.6.4 L1 data memory system

The L1 data cache is organized as a *Virtually Indexed Physically Tagged* (VIPT) cache, with alias avoidance logic so that it appears to software as if it were physically indexed.

The Arm®v8-A architecture does not support an operation to invalidate the entire data cache. If software requires this function, it must be constructed by iterating over the cache geometry and executing a series of individual invalidate by set/way instructions.

### 2.6.4.1 Memory system implementation

This section describes the implementation of the L1 memory system.

#### Limited Order Regions

The Cortex®-A65AE core supports four limited order ranges that include the entire memory space.

#### Atomic instructions

The Cortex®-A65AE core supports the atomic instructions added in the Arm®v8.1-A architecture.

Atomic instructions to cacheable memory can be performed as either near atomics or far atomics, depending on where the cache line containing the data resides. If the instruction hits in the L1 data cache in a unique state then it will be performed as a near atomic in the L1 memory system. If the atomic operation misses in the L1 cache, or the line is shared with another core then the atomic is sent as a far atomic out to the L3 cache. If the operation misses everywhere within the cluster, and the master interface is configured as CHI, and the interconnect supports far atomics, then the atomic will be passed on to the interconnect to perform the operation. If the operation hits anywhere inside the cluster, or the interconnect does not support atomics, then the L3 memory system will perform the atomic operation and allocate the line into the L3 cache if it is not already there.

The Cortex®-A65AE core supports atomics to device or non-cacheable memory, however this relies on the interconnect also supporting atomics. If such an atomic instruction is executed when the interconnect does not support them, it will result in a synchronous Data Abort (for load atomics) or an asynchronous Data Abort (for store atomics). The behavior of the atomic instructions can be modified by the CPUECTLR register settings.

For more information on the CPUECTLR register, see [3.1.25 CPUECTLR\\_EL1, CPU Extended Control Register, EL1](#) on page 121.

## LDAPR instructions

The core supports Load acquire instructions adhering to the RCpc consistency semantic introduced in the Arm®v8.3-A extensions. This is reflected in register ID\_AA64ISAR1\_EL1 where bits[23:20] are set to 0b0001 to indicate that the core supports LDAPRB, LDAPRH, and LDAPR instructions implemented in AArch64.

For more information on the ID\_AA64ISAR1\_EL1 register, see [3.1.53 ID\\_AA64ISAR0\\_EL1, AArch64 Instruction Set Attribute Register 0, EL1](#) on page 152.

## 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.

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.

For streams of contiguous stores that are targeted at a memory region that is marked as transient, the following occurs:

- A line may or may not be allocated if the stores coalesce a full cache line and miss the L1 data cache, the cache line is streamed to the external memory system without being allocated into the L1 data cache, the L2 cache, or the L3 cache.
- A line may or may not be allocated if the stores do not cover the entire cache line and miss the L1 data cache, the modified cache line will not allocate into the L1 data cache but will allocate and write into the L2 cache, marking it as transient.

## 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.

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 cacheline 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 L1, is not allocated to L2 as would happen for a normal line.



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

---

Non-temporal stores are treated as if write streaming mode was active. They are not allocated into any cache in the cluster unless they hit in the cache.

## 2.6.4.2 Internal exclusive monitor

The Cortex®-A65AE core L1 memory system has internal exclusive monitors, one for each thread.

The exclusive monitor for each thread is a 2-state, open and exclusive, state machine that manages Load-Exclusive or Store-Exclusive accesses and Clear-Exclusive (`CLREX`) instructions. You can use these instructions to construct semaphores, ensuring synchronization between different processes running on the core, and also between different cores that are using the same coherent memory locations for the semaphore. A Load-Exclusive instruction tags a small block of memory for exclusive access. `CTR.ERG` defines the size of the tagged block as 16 words, one cache line.



A load/store exclusive instruction is any 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 SoC that does not support exclusive accesses, it causes a Data Abort exception with a Data Fault Status Code of either:

- `0b110101`, when using the long descriptor format.
- `0b10101`, when using the short descriptor format.

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

### 2.6.4.2.1 Treatment of intervening STR operations

Where there is an intervening store operation between an exclusive load and an exclusive store from the same thread, the intermediate store does not produce any direct effect on the internal exclusive monitor.

After the exclusive load, the local monitor is in the Exclusive Access state. It remains in the Exclusive Access state after the store, and then returns to the Open Access state only after an exclusive store, a `CLREX` instruction, or an exception return.

However, if the exclusive code sequence accessed address is in cacheable memory, any eviction of the cache line containing that address clears the monitor. Arm recommends that no load or store instructions are placed between the exclusive load and the exclusive store, because these additional instructions can cause a cache eviction. Any data cache maintenance instruction can also clear the exclusive monitor.



A store from the opposite thread in the same core will clear the internal exclusive monitor.

### 2.6.4.3 Exclusive monitor

In the exclusive state machine, the **IMPLEMENTATION DEFINED** transitions are as follows:

- If the monitor is in the exclusive state, and a store exclusive is performed to a different address, then the store exclusive fails and does not update memory.
- If the monitor is in the exclusive state, and a store exclusive is performed on the same thread to a different address, then the store exclusive will:
  - Fail.
  - Clear the local exclusive monitor.
  - Not update the memory.
  - Not clear the global monitor.
  - Not send a wakeup event.

In this case, clearing the global monitor and sending the wakeup event is **IMPLEMENTATION DEFINED**.

- 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 thread to the same address it clears the exclusive monitor. If the store is from the same thread then it does not clear the monitor.

### 2.6.5 Data prefetching

The following section describes the software and hardware data prefetching behavior of the Cortex®-A65AE core.

#### Hardware data prefetcher

The Cortex®-A65AE core has a data prefetch mechanism that looks for cache line fetches with regular patterns. If the data prefetcher detects a pattern, then 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.

The Cortex®-A65AE core can track multiple streams in parallel.

Prefetch streams end when either:

- The pattern is broken.
- A DSB is executed.
- A WFI or WFE is executed.
- A data cache maintenance operation is executed.

For read streams, the prefetcher is based on the virtual addresses. A given stream is allowed to prefetch addresses through multiple pages as long as they are cacheable and with read permissions. If the new page is still cacheable and has read permission, it can cross page boundaries. Write



streams are based on physical addresses and so cannot cross page boundaries. However, if full cache line writes are performed then the prefetcher does not activate and write streaming mode is used instead.

For some types of pattern, when the prefetcher is confident in the stream, it can start progressively increasing the prefetch distance ahead of the current accesses. These accesses start to allocate to the L3 cache rather than L1. Allocating to the L3 cache allows better utilization of the larger resources available at L3. Also, utilizing the L3 cache reduces the amount of pollution of the L1 cache if the stream ends or is incorrectly predicted. If the prefetching to L3 was accurate, the line will be removed from L3 and allocated to L1 when the stream reaches that address.

The CPUECTLR register allows you to:

- Deactivate the prefetcher.
- Alter the number of outstanding requests that the prefetcher can make.

### Preload instructions

The Cortex®-A65AE core supports `PRFM` instructions. If `PRFM` miss and are to a cacheable address, then these instructions perform a lookup in the cache and start a linefill. A prefetch request can be sent to L2 or L3 to start a linefill, and then the instruction can retire without any data being returned to L1. `PLI`, `PLIL1KEEP`, and `PLIL1STRM` are implemented as a prefetch to L2.

Use the `PRFM` instruction for data prefetching where short sequences or irregular pattern fetches are required. For more information about prefetch memory and preloading caches, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### Data Cache Zero

The Data Cache Zero by Virtual Address (`DC ZVA`) instruction enables a block of 64-bytes in memory, which is aligned to 64-bytes in size, to be set to 0. The `DCZID_ELO` register passes this value.

The `DC ZVA` instruction allocates this value into the data cache using the same method as a normal store instruction.

## 2.6.6 Direct access to internal memory

The Cortex®-A65AE core provides a mechanism to read the internal memory that is used by the L1 cache and TLB structures through implementation defined system registers. This functionality can be useful when investigating issues where the coherency between the data in the cache and data in system memory is broken.

The appropriate memory block and location are selected using several write-only registers. The data is read from read-only registers as shown in the following table. These operations are available only in EL3. In all other modes, executing these instructions results in an Undefined Instruction exception.

**Table 2-11: AArch64 registers used to access internal memory**

Register name	Function	Access	Operation	Rd Data
CDBGDR0_EL3	Data Register 0	Read-only	MRS <Xd>, S3_3_c15_c0_0	Data
CDBGDR1_EL3	Data Register 1	Read-only	MRS <Xd>, S3_3_c15_c0_1	Data
CDBGDR2_EL3	Data Register 2	Read-only	MRS <Xd>, S3_3_c15_c0_2	Data
CDBGDCT_EL3	Data Cache Tag Read Operation Register	Write-only	sys #6, c15, c2, #0, <Xd>	Set/Way
CDBGICT_EL3	Instruction Cache Tag Read Operation Register	Write-only	sys #6, c15, c2, #1, <Xd>	Set/Way
CDBGTT_EL3	TLB Tag Read Operation Register	Write-only	sys #6, c15, c2, #2, <Xd>	Index/Way
CDBGDCD_EL3	Data Cache Data Read Operation Register	Write-only	sys #6, c15, c4, #0, <Xd>	Set/Way/Offset
CDBGICD_EL3	Instruction Cache Data Read Operation Register	Write-only	sys #6, c15, c4, #1, <Xd>	Set/Way/Offset
CDBGTD_EL3	TLB Data Read Operation Register	Write-only	sys #6, c15, c4, #2, <Xd>	Index/Way

### 2.6.6.1 Encoding for tag and data in the L1 data cache

The Cortex®-A65AE L1 data cache is a 4-way set associative structure.

The size of the configured cache determines the number of sets in each way. The following table shows the encoding (set in *ra* in the appropriate *mcr* instruction) used to locate the cache data entry for tag and data memory. It is similar for both the tag and data RAM access.

Data RAM access includes an extra field to locate the appropriate word in the cache line. The set-index range parameter (*S*) is:

S=13      For a 32KB cache.  
S=14      For a 64KB cache.

**Table 2-12: Cortex®-A65AE L1 Data Cache Tag and Data location encoding**

Bitfield of Rd	Description
[31:30]	Cache Way
[29:S]	Unused
[S-1:6]	Set index
[5:3]	Cache data element offset
[2:0]	Unused (Zero)

Tag information (MESI state, outer attributes, and valid) for the selected cache line, returns using Data Register 0 and Data Register 1.

Use the format that is shown in the following table.

**Table 2-13: Cortex®-A65AE L1 Data Cache Tag data format**

Bitfield of Data Register 0 and 1	Description
DR1[31:25]	UNKNOWN

Bitfield of Data Register 0 and 1	Description
DR1[24:23]	MESI State (from tag RAM): 0b00 Invalid 0b01 Shared 0b10 Unique non-transient 0b11 Unique transient
DR1[22]	Non-secure state (NS) (from tag RAM)
DR1[21:0]	Tag Address [43:22] (from tag RAM)
DR0[31:22]	Tag Address [21:12] (from tag RAM)
DR0[21:9]	Unused (Zero)
DR0[8:5]	UNKNOWN
DR0[4]	Dirty bit (from Dirty RAM)
DR0[3]	Shareability (from Dirty RAM)
DR0[2:1]	Age (from Dirty RAM)
DR0[0]	Outer Allocation Hint (from Dirty RAM)

The 64 bits of cache data returns in Data register 0 and Data register 1.

### 2.6.6.2 Encoding for tag and data in the L1 instruction cache

The L1 instruction cache is different from the L1 data cache. This is shown in the encodings and data format used in the cache debug operations that are used to access the tag and data memories.

The following table shows the encoding that is required to select a given cache line.

The set-index range parameter (S) is:

S=13 For a 32KB cache.  
S=14 For a 64KB cache.

**Table 2-14: Cortex®-A65AE Instruction Cache Tag and Data location encoding**

Bitfield of Rd	Description
[31:30]	Cache Way
[29:S]	Unused
[S-1:6]	Set index
[5:2]	Cache data element offset (Data Register only)
[1:0]	Unused

The following table shows the tag, instruction, and valid data for the selected cache line using only Data Register.

**Table 2-15: Cortex®-A65AE Instruction Cache Tag data format**

Bitfield of Data Register 0	Description
[31]	Unused
[30:29]	Valid and set mode:
	0b10 A64
	0b11 Invalid
[28]	Non-secure state (NS) -
[27:0]	Tag address -

The cache data RAMs store instructions in a pre-decoded format. The L1 Instruction Cache Data Read Operation returns two 20-bit entries from the cache in Data Register 0. Each corresponds to the 16-bit aligned offset in the cache line:

Data Register 0[19:0] Pre-decode data from cache offset.

Data Register 1[19:0] Pre-decode data from cache offset +2.

### 2.6.6.3 Encoding for the L2 TLB

The Cortex®-A65AE core L2 TLB is built from a 4-way set associative RAM-based structure and contains the data for the main TLB RAM, the Walk cache and IPA cache.

To read the individual entries into the data registers, software must write to the TLB Tag Read Operation Register and to the TLB Data Read Operation Register.

**Table 2-16: Cortex®-A65AE TLB Data Read Operation Register location encoding**

Bitfield of Rd	Description
[31:30]	TLB Way
[29:9]	Unused
[8:0]	TLB index

The TLB index is used to select the index from the TLB, walk cache, or IPA cache.

**Table 2-17: TLB index**

Bitfield of Rd	Description
0x000-0FF	Main TLB
0x100-10F	Walk cache
0x110-11F	IPA cache

The TLB uses an encoding for the descriptor that is returned using the following Data Registers:

Data Register 0[31:0] TLB Descriptor[31:0]  
 Data Register 1[31:0] TLB Descriptor[63:32]  
 Data Register 2[31:0] TLB Descriptor[88:64]

#### 2.6.6.4 Main TLB RAM descriptor fields

The Main TLB RAM is divided into two parts, where one part for storing the tag and the other for storing the data. The following tables list the descriptor fields.

**Table 2-18: TLB descriptor fields for Tag RAM**

Field	Bits	Width	Description
Valid	[0]	1	Indicates that the entry is valid.
NS (walk)	[1]	1	The security state of core. Used to compare with the NS state for TLB lookup entry match.
ASID	[17:2]	16	Indicates the <i>Address Space Identifier</i> (ASID). This field will be 0 if ASID is not used.
VMID	[33:18]	16	Indicates the virtual machine identifier. This field will be 0 if VMID is not used.
Thread ID	[34]	1	Indicates the Thread ID for the software thread that wrote the translation descriptor in the main TLB RAM.
Size[1:0]	[36:35]	2	Indicates the combined page size of stage 1 and stage 2.  The Domain[1] bit (bit[44]) is used together to encode the size information as a 3-bit field combination {Domain[1]:Size[1:0]}:  0b000      4KB 0b001      64KB 0b010      2MB 0b011      512MB 0b100      16KB 0b101      32MB
nG	[37]	1	Indicates the non-global bit.
AP/HYP	[40:38]	3	Indicates that stage 1 access permission is represented by the S2AP field for either: <ul style="list-style-type: none"> <li>EL2/EL3 translation regimes.</li> <li>EL0 translation regimes, which are non-secure when the <i>Virtual Host Extension</i> (VHE) is configured.</li> </ul>
S2AP	[42:41]	2	When VHE is not configured, S2AP indicates the stage 2 permission for EL1/ELO. When VHE is configured, S2AP[0] indicates that the cached translation applies to EL3/EL2/ELO. Additionally, S2AP[1] represents stage 1 access permission.
Domain	[46:43]	4	Indicates miscellaneous control information like Common not Private bit state, AP[1] access permission for ELO in VHE or stage 1 Dirty Bit Modifier (DBM) bit when not in VHE mode, or last level of translation table walk as Level 3 or Level 2 for each translation granule size.
S1 Size	[49:47]	3	Indicates the page or block size of the stage 1 translation result.
Address Sign bit	[50]	1	Indicates the VA sign bit, VA[48].

Field	Bits	Width	Description
VA	[78:51]	28	Indicates the virtual address.
DBM	[79]	1	Indicates the <i>Dirty Bit Modifier</i> (DBM) bit for stage 1 translation table walks when in VHE mode or stage 2 translation table walks when not in VHE mode.
Parity	[81:80]	2	Indicates the parity bits.

**Table 2-19: TLB descriptor fields for Data RAM**

Field	Bits	Width	Description
XS1Usr	[0]	1	Executable in stage 1 user mode
XS1Non-Usr	[1]	1	Executable in stage 1 non-user mode
XS2Usr	[2]	1	Executable in stage 2 user mode
XS2Non-Usr	[3]	1	Executable in stage 2 non-user mode
Memory type and shareability	[11:4]	8	Defines the memory attribute
S2 Level	[13:12]	2	The stage 2 level that gave this translation
NS (descriptor)	[14]	1	The security state allocated to this memory region
PA	[46:15]	32	The physical address
Parity	[47]	1	Parity

### 2.6.6.5 Walk cache descriptor fields

The following table shows the walk cache descriptor data fields for Tag and Data RAMs.

**Table 2-20: Walk cache descriptor fields for Tag RAM**

Field	Bit Position	Width	Description
Valid	[0]	1	Indicates that the entry is valid
NS (walk)	[1]	1	The Security state of the entry fetch
ASID	[17:2]	16	Address Space Identifier
VMID	[33:18]	16	Virtual Machine Identifier
HYP/EL2	[34]	1	Set if the entry was fetched in HYP, EL2, or <i>Virtual Host Extension</i> (VHE) mode.
EL3	[35]	1	Set if the entry was fetched in AArch64 EL3 mode
Arch	[38:36]	3	Used to determine how many and which bits of address are used for constructing the physical address of the pagewalk
Domain	[42:39]	4	Invalid
Thread ID	[43]	1	Thread ID for the thread that wrote the translation descriptor in Walk cache
CnP	[44]	1	Common not Private bit
Address Sign Bit	[45]	1	Address sign bit, VA[48]
VA	[69:46]	24	Virtual Address sign bit
S2AP	[71:70]	2	Stage 2 access permission
S2level	[73:72]	2	The stage 2 level which translates the IPA to PA for the page table entry
Parity	[81:80]	2	Parity Bits

**Table 2-21: Walk cache descriptor fields for Data RAM**

Field	Bit Position	Width	Description
APTable	[1:0]	2	Combined ATable bits from stage 1 descriptors up to the last level
XNTable	[2]	1	Combined XNTable bits from stage 1 descriptors up to the last level
PXNTable	[3]	1	Combined PXNTable bits from stage 1 descriptors up to the last level
NSTable	[4]	1	Combined NSTable bits from first and second-level stage 1 tables or NS descriptors (VMSA)
Attrs	[12:5]	8	Physical address attributes of the final level stage 1 table
PA	[46:13]	34	Physical address of the stage 1 last translation level page table entry
Parity	[47]	1	Parity

### 2.6.6.6 IPA cache descriptor fields

The IPA cache holds mappings from intermediate physical addresses (IPA) to physical addresses. It is only used for translations performed in non-secure ELO/1. It is updated whenever a stage 2 translation is completed, and checked whenever a stage 2 translation is required.

The following table shows the data and tag fields in the IPA cache descriptor.

**Table 2-22: IPA cache descriptor fields for Tag RAM**

Fields	Bits	Width	Descriptor
Valid	[0]	1	Indicates that the entry is valid.
Entry granule	[2:1]	2	Indicates the entry granule size.
Thread ID	[3]	1	Indicates Thread ID.
CnP	[4]	1	Indicates the Common not Private bit.
Size	[8:5]	4	Indicates the S2 page size for this entry.
DBM	[9]	1	Indicates the DBM.
Unused	[17:10]	8	Must be set to 0.
VMID	[33:18]	16	Indicates the virtual machine identifier.
IPA	[61:34]	28	Unused lower bits, page size dependant, must be set to zero.
Unused	[79:62]	18	Must be set to 0.
Parity	[81:80]	2	Parity.

**Table 2-23: IPA cache descriptor fields for Data RAM**

Fields	Bits	Width	Descriptor
SH	[1:0]	2	Shareability.
S2AP	[3:2]	2	Stage 2 access permissions
XN	[5:4]	2	Controls EL1 and ELO access permissions.
Memattrs	[9:6]	4	Stage 2 memory attributes.
PA	[41:10]	32	Physical Address.
Unused	[46:42]	5	Must be set to 0.
Parity	[47]	1	Parity.

## 2.7 L2 memory system

This chapter describes the L2 memory system.

### 2.7.1 About the L2 memory system

The Cortex®-A65AE L2 memory system is required to interface the Cortex®-A65AE cores to the L3 memory system.

The L2 cache controller 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, which can then take precise or imprecise aborts, depending on the type of transaction.

The L2 memory subsystem consists of:

- An optional 4-way, set-associative L2 cache with a configurable size of 64KB, 128KB, or 256KB. Cache lines have a fixed length of 64 bytes.
- ECC protection for tag, data, and L2 data buffer RAM structures.

The main features of the L2 memory system are:

- Strictly exclusive with L1 data cache.
- Pseudo-inclusive with L1 instruction cache.
- Private per-core unified L2 cache.
- 44-bit physical address space.
- Physically indexed, physically tagged.
- In Lock-mode, L2 tag and data pipelines are forced to inline correction mode to allow single-bit ECC error correction without causing lock-step divergence.

### 2.7.2 Optional integrated L2 cache

Data is allocated to the L2 cache only when evicted from the L1 memory system, not when first fetched from the system.

The exceptions to this rule are:

- If the Read-Allocate hint is set, cacheable reads from the TLB or instruction side are allocated in the L2 cache.
- If the Write-Allocate hint is set when the L1 enters write-streaming mode, cacheable writes are allocated in the L2 until the L2 streaming threshold is reached.
- L2 prefetches issued by the L1 through a `PRFM` instruction are allocated in the L2 regardless of the Read-Allocate hint.



When non-temporal data is evicted from the L1 memory system, the data is sent directly to L3 and is not allocated in L2.

L2 RAMs are invalidated automatically at reset unless the debug recovery P-Channel state is used.

### 2.7.3 Support for memory types

The Cortex®-A65AE 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 Non-cacheable.

The additional attribute hints are used as follows:

#### Allocation hint

Determines the rules of allocation of newly fetched lines in the system, see [2.7.2 Optional integrated L2 cache](#) on page 72.

#### Transient hint

Allocating reads (from TLB or instruction side) and writes in write-streaming mode that have the transient bit set are allocated in L2 cache and marked as most likely to be evicted according to the L2 eviction policy.

Evictions from L1 cache marked as transient are not allocated in L2 cache.

The standard CHI attributes are passed to DSU-AE with no modifications (except for translating architectural attributes to CHI attributes):

- Allocate hint.
- Cacheability (inner and outer are merged together, as the Cortex®-A65AE core only allocates both inner and outer cacheable memory).
- Shareability.

## 2.8 Reliability, Availability, and Serviceability (RAS)

This chapter describes the RAS features implemented in the Cortex®-A65AE core.

## 2.8.1 Cache ECC and parity

The Cortex®-A65AE core implements the *Reliability, Availability, and Serviceability* (RAS) extension to the Arm®v8-A architecture which provides mechanisms for standardized reporting of the errors that are generated by cache protection mechanisms.

Core cache protection enables the Cortex®-A65AE core to detect and correct a 1-bit error in any RAM and detect 2-bit errors in some RAMs.



For information about SCU-L3 cache protection, see the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual*.

---

The RAS extension improves the system by reducing unplanned outages:

- Transient errors can be detected and corrected before they cause application or system failure.
- Failing components can be identified and replaced.
- Failure can be predicted ahead of time to allow replacement during planned maintenance.

The severity of a failure can range from minor to catastrophic. In many systems, data or service loss is regarded as more of a minor failure than data corruption, as long as backup data is available.

The RAS extension focuses on errors that are produced from hardware faults, which fall into two main categories:

- Transient faults.
- Persistent faults.

The RAS extension describes data corruption faults, which mostly occur in memories and on data links. RAS concepts can also be used for the management of other types of physical faults that are found in systems, such as lock-step errors, thermal trip, and mechanical failure. The RAS extension provides a common programmers model and mechanisms for fault handling and error recovery.

## 2.8.2 Cache protection behavior

The core protects against soft errors that result in a RAM bitcell temporarily holding the incorrect value.

The Cortex®-A65AE core writes a new value to the RAM to correct the error. If the error is a hard error that is not corrected by writing to the RAM, for example a physical defect in the RAM, and if there are two or more than two such hard errors in the core, then the core might get into a livelock as it continually detects and then tries to correct the error.

Some RAMs have *Single Error Detect* (SED) capability, while others have *Single Error Correct, Double Error Detect* (SECCDED) capability. The core can make progress and remain functionally correct when there is transient single bit error in any RAM. If there are multiple single bit errors in different

RAMs, or within different protection granules within the same RAM, then the core also remains functionally correct. If there is a double bit error in a single RAM within the same protection granule, then the behavior depends on the RAM:

- For RAMs with SECDED capability listed in the following table, the error is detected and reported as described in error reporting. If the error is in a cache line containing dirty data, then that data might be lost, resulting in data corruption
- For RAMs with only SED, a double bit error is not detected and therefore might cause data corruption.

If there are three or more bit errors, then depending on the RAM and the position of the errors within the RAM, the errors might be detected or might not be detected.

The Cortex®-A65AE cache protection support has a minimal performance impact when no errors are present. When an error is detected, the access that caused the error is stalled while the correction takes place. When the correction is complete, the access either continues with the corrected data, or is retried. If the access is retried, it either hits in the cache again with the corrected data, or misses in the cache and re-fetches the data from a lower level cache or from main memory. The behavior for each RAM is shown in the following table.

**Table 2-24: Cache protection behavior**

RAM	Protection type	Protection granule	Correction behavior
L1 instruction cache tag	Parity, SED	34 bits	Both lines in the cache set are invalidated, then the line requested is refetched from L2 or external memory.
L1 instruction cache data	Parity, SED	32 bits	Both lines in the cache set are invalidated, then the line requested is refetched from L2 or external memory.
L2 TLB tag	Parity, SED	40 bits	Entry invalidated, new pagewalk started to refetch it.
L2 TLB data	Parity, SED	47 bits	Entry invalidated, new pagewalk started to refetch it.
L1 data cache tag	ECC, SECDED	35 bits	Line cleaned and invalidated from L1. SCU duplicate tags are used to get the correct address. Line refetched from L2 or external memory, with single bit errors corrected as part of the eviction.
L1 data cache data	ECC, SECDED	32 bits	Line cleaned and invalidated from L1, with single bit errors corrected as part of the eviction. Line refetched from L2 or external memory.
L1 data cache dirty	ECC, SECDED	2 bits	Line cleaned and invalidated from L1, with single bit errors corrected as part of the eviction. Only the dirty bit is protected. The other bits are performance hints, therefore do not cause a functional failure if they are incorrect.
L2 cache tag	ECC, SECDED	30, 31, or 32 bits depending on the cache size.	Tag rewritten with correct value, access retried. If the error is uncorrectable then the tag is invalidated.
L2 cache victim	None	-	The victim RAM is used only as a performance hint. It does not result in a functional failure if the contents are incorrect.

RAM	Protection type	Protection granule	Correction behavior
L2 cache data	ECC, SECEDED	64 bits	<ul style="list-style-type: none"> <li>In Split-mode, an additional cycle is required for correction if an error is detected.</li> <li>In Lock-mode, an additional cycle for inline correction is always needed by both cores in order for the two cores in a core pair to remain in lock-step.</li> </ul>
L2 data buffer	ECC, SECEDED	72 bits	<ul style="list-style-type: none"> <li>In Split-mode, an additional cycle is required for correction if an error is detected.</li> <li>In Lock-mode, an additional cycle for inline correction is always needed by both cores in order for the two cores in a core pair to remain in lock-step.</li> </ul>
Branch predictor	None	-	The branch predictor RAMs are used only as a performance hint. They do not result in a functional failure if the contents are incorrect.

### Persistent error handling

A persistent, hard, or stuck-at RAM error is an error that cannot be corrected. Execution on the Cortex®-A65AE core with instruction, data, and translation table RAMs sourced from non-cacheable memory will not livelock due to any single persistent errors in the:

- L1 instruction cache.
- L1 data cache.
- Main TLB cache.
- L2 cache.
- L3 cache.

Cache coherency is not guaranteed past a detected persistent error.

## 2.8.3 Uncorrected errors and data poisoning

When an error is detected, the correction mechanism is triggered. However, if the error is a 2-bit error in a RAM protected by ECC, then the error is not correctable.

The behavior on an uncorrected error depends on the type of RAM.

### Uncorrected error detected in a data RAM

When an uncorrected error is detected in a data RAM:

- The chunk of data with the error is marked as poisoned. This poison information is then transferred with the data and stored in the cache if the data is allocated back into a cache. The poisoned data is stored per 64 bits of data, except in the L1 data cache where it is stored per 32 bits of data.
- If the interconnect supports poisoning, then the poison is passed along with the data when the line is evicted from the cluster. No abort is generated when a line is poisoned, as the abort can be deferred until the point when the poisoned data is consumed by a load or instruction fetch.

### Uncorrected error detected in a tag or dirty RAM

When an uncorrected error is detected in a tag RAM or dirty RAM, either the address or coherency state of the line is not known anymore, and the data cannot be poisoned. In this case, the line is invalidated and an interrupt is generated to notify software that data has potentially been lost.

## 2.8.4 RAS error types

For a standard error record, three error types can be recorded.

When a processing element accesses memory or other state, errors might be detected in that memory or state, and corrected, deferred, or signaled to the processing element as a detected error. The component that detects an error is called a node.

### Corrected error (CE)

An error was detected and corrected. It no longer infects the state of the node and has not been silently propagated. The node continues to operate.

### Deferred error (DE)

An error was detected, was not corrected, and was deferred. The error is not silently propagated and may be latent in the system. The node continues to operate.

### Uncorrected Error (UC)

An error was detected and was not corrected or deferred. The error is latent in the system.



Uncorrected errors can have subtypes depending on whether the error was produced or consumed at the node.

---

### Errors produced at the node

For uncorrected errors that are produced at the node, the subtypes, in increasing severity, are:

#### Latent

The error has not been propagated. That is, the error was detected but not consumed, and was not recorded as a deferred error.

#### Signaled

The error has not been silently propagated. The error has been or might have been consumed, and was not recorded as a deferred error.



The producer cannot know if a consumer has architecturally consumed the error. If it has definitely not been propagated to any consumer, and signaled otherwise, an error might be marked as Latent.

---

### Unrecoverable (UEU)

The error has not been silently propagated. The node cannot continue operating.

### Uncontainable (UC)

The error might have been silently propagated. If the error cannot be isolated, the system must be shut down to avoid catastrophic failure.

## Errors consumed at the node

For uncorrected errors that are consumed at the node, the subtypes, in increasing severity, are:

### Restartable (UEO)

The error has not been silently propagated. The node halts operation because of consuming an error. The node does not rely on the corrupted data so can continue to operate without repairing the error.

### Recoverable (UER)

The error has not been silently propagated. The node halts operation. To continue, the node relies on consuming the corrupted data. If software can locate and repair the error, the halted operation can continue.

### Unrecoverable (UEU)

The error has not been silently propagated. The node halts operation and cannot resume from its halted state.

### Uncontainable (UC)

The error might have been silently propagated. If the error cannot be isolated, the system must be shut down to avoid catastrophic failure.

## Error status priority

The highest priority recorded error type is recorded in the Selected Error Record Primary Status Register.

For more information, see [3.2.8 ERROSTATUS, Error Record Primary Status Register](#) on page 202.

## Related information

[ERXSTATUS\\_EL1, Selected Error Record Primary Status Register, EL1](#) on page 145

## 2.8.5 Error synchronization barrier

The *Error Synchronization Barrier* ( $\text{ESB}$ ) instruction synchronizes unrecoverable errors.

The RAS extension adds the  $\text{ESB}$  instruction used to synchronize unrecoverable errors. Unrecoverable errors are containable errors consumed by the core and not silently propagated.

The  $\text{ESB}$  instruction allows efficient isolation of errors:

- The  $\text{ESB}$  instruction does not wait for completion of accesses that cannot generate an asynchronous external abort. For example, if all external aborts are handled synchronously or it is known that no such accesses are outstanding.
- The  $\text{ESB}$  instruction does not order accesses and does not guarantee a pipeline flush.

All unrecoverable errors must be synchronized by an  $\text{ESB}$  instruction. The  $\text{ESB}$  instruction guarantees the following:

- All unrecoverable errors that are generated before the  $\text{ESB}$  instruction have pended a *System Error Interrupts* (SEI) exception.

- If a physical SEI is pending by or was pending before the `ESB` instruction is executed:
  - If the physical SEI is unmasked at the current Exception level, then it is taken before completion of the `ESB` instruction.
  - If the physical SEI is masked at the current Exception level, the pending SEI is cleared, the SEI syndrome is recorded in `DISR/DISR_EL1`, and `DISR/DISR_EL1.A` is set to 1. This indicates that the SEI was generated before the `ESB` by instructions that occur in program order.

The `ESB` instruction also guarantees the following:

- SEIs generated before the `ESB` instruction are either taken before or at the `ESB` instruction, or are pending in `DISR/DISR_EL1`.
- SEIs generated after the `ESB` are not pending in `DISR/DISR_EL1`.

This includes unrecoverable errors that are generated by instructions, translation table walks, and instructions fetches on the same core.

---

`DISR_EL1` can only be accessed at EL1 or above. If EL2 is implemented and `HCR_EL2.AMO` is set to 1, then reads and writes of `DISR_EL1` at Non-secure EL1 access `VDISR_EL2`.



Note

See the following registers:

- [3.1.34 DISR\\_EL1, Deferred Interrupt Status Register, EL1](#) on page 137.
  - [3.1.48 HCR\\_EL2, Hypervisor Configuration Register, EL2](#) on page 148.
  - [3.1.81 VDISR\\_EL2, Virtual Deferred Interrupt Status Register, EL2](#) on page 187.
- 

## 2.8.6 Error recording

The component that detects an error is called a node. The Cortex®-A65AE core is a node that interacts with the DynamIQ™ Shared Unit AE node. There is one record per node for the errors detected.

For more information on error recording that is generated by cache protection, see the *Arm® Reliability, Availability, and Serviceability (RAS) Specification, Armv8, for the Armv8-A architecture profile*. The following points apply specifically to the Cortex®-A65AE core:

- In the Cortex®-A65AE core, any error that is detected is reported and recorded in the error record registers:
  - [3.1.36 ERRSELR\\_EL1, Error Record Select Register, EL1](#) on page 139
  - [3.1.37 ERXCTLR\\_EL1, Selected Error Record Control Register, EL1](#) on page 140
  - [3.1.38 ERXFR\\_EL1, Selected Error Record Feature Register, EL1](#) on page 140
  - [3.1.39 ERXMISCO\\_EL1, Selected Error Record Miscellaneous Register 0, EL1](#) on page 141
  - [3.1.40 ERXPFGCDN\\_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1](#) on page 141
  - [3.1.41 ERXPFGCTL\\_EL1, Selected Error Pseudo Fault Generation Control Register, EL1](#) on page 142
  - [3.1.42 ERXPFGF\\_EL1, Selected Pseudo Fault Generation Feature Register, EL1](#) on page 144
  - [3.1.43 ERXSTATUS\\_EL1, Selected Error Record Primary Status Register, EL1](#) on page 145
- There are two error records provided, which can be selected with the ERRSELR\_EL1 register:
  - Record 0 is private to the core, and is updated on any error in the core RAMs including L1 caches, TLB, and L2 cache.
  - Record 1 records any error in the L3 and snoop filter RAMs and is shared between all cores in the cluster.
- The fault handling interrupt is generated on the **nFAULTIRQ[0]** pin for L3 and snoop filter errors, or on the **nFAULTIRQ[n+1]** pin for core *n* L1 and L2 errors.



The Cortex®-A65AE core does not provide the ERXADDR\_ELA, Selected Error Record Address Register, EL1.

## 2.8.7 Error injection

The Cortex®-A65AE core supports fault injection for the purpose of testing fault handling software.

The core is programmable to inject an error for any of the possible error types (corrected error, deferred error, uncontrollable error, and recoverable error) on a future memory access. When that access is performed, the core responds as if an error was detected on that access by asserting error interrupts, logging information in the error records, and taking aborts as appropriate for the type of error. Injecting an error will not affect the data in the RAM or the checking process itself. When a real error is detected on an access for which an injected error is programmed, the injected error will not prevent the core from handling the real error. The RAS register might log the injected error or the real error in this case.

To get the error injection to work:



- Program the Error Record Select Register (ERRSELR\_EL1) to select Error record 0.
- Program the Error Record Control Register (ERROCTLR) to enable error detection/recovery and fault detection.
- Program the Error Pseudo Fault Generation Control Register (ERROPFGCTL) to allow error injection.



Cacheable code must also be executed, which will cause Cacheable transactions that can be injected with errors.

The following table describes the possible types of errors that the core can encounter and therefore inject.

**Table 2-25: Errors injected in the Cortex®-A65AE core**

Error type	Description
Corrected errors	A CE is generated for a single ECC error on L1 data cache access.
Deferred errors	A DE is generated for a double ECC error on eviction of a cache line from the L1 to the L2, or as a result of a snoop on the L1.
Uncontainable errors	A UC is generated for a double ECC error on the L1 TAG RAM following an eviction.
Recoverable error	A UER is generated as a recoverable error.

## Error registers

The following table describes the registers that handle error injection in the Cortex®-A65AE core.

**Table 2-26: Error injection registers**

Register name	Description
ERROPFGF	The ERR Pseudo Fault Generation Feature register defines which errors can be injected.
ERROPFGCTL	The ERR Pseudo Fault Generation Control register controls the errors that are injected.
ERROPFGCDN	The Selected Pseudo Fault Generation Count Down register controls the fault injection timing.



This mechanism simulates the corruption of any RAM but the data is not actually corrupted.

See also:

- [3.2.7 ERROPFGF, Error Pseudo Fault Generation Feature Register](#) on page 200.
- [3.2.6 ERROPFGCTL, Error Pseudo Fault Generation Control Register](#) on page 198.
- [3.2.5 ERROPFGCDN, Error Pseudo Fault Generation Count Down Register](#) on page 198.

## 2.9 Generic Interrupt Controller CPU interface

This chapter describes the Cortex®-A65AE core implementation of the Arm *Generic Interrupt Controller* (GIC) CPU interface.

### 2.9.1 About the Generic Interrupt Controller CPU interface

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

The GIC CPU interface hosts registers to mask, identify, and control states of interrupts forwarded to that core. There is a separate GIC CPU interface for each core in the system.

The Cortex®-A65AE core implements the GIC CPU interface as described in the *Arm® Generic Interrupt Controller Architecture Specification*. This interfaces with an external GICv3 or GICv4 interrupt distributor component within the system.



This chapter describes only features that are specific to the Cortex®-A65AE core implementation. Additional information specific to the DSU-AE can be found in *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual*.

---

The GICv4 architecture supports:

- Two security states.
- Interrupt 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.
- Wake-up events in power management environments.

The GIC includes interrupt grouping functionality that supports:

- Configuring each interrupt to belong to an interrupt group.
- Signaling Group 1 interrupts to the target core using either the IRQ or the FIQ exception request.
- 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.

This chapter describes only features that are specific to the Cortex®-A65AE core implementation.

## 2.9.2 Bypassing the CPU interface

The GIC CPU interface is always implemented within the Cortex®-A65AE core.

However, you can disable it if you assert the **GICCDISABLE** signal HIGH at reset. If the GIC is enabled, the input pins **nVIRQ** and **nVFIQ** must be tied off to HIGH. This is because the internal GIC CPU interface generates the virtual interrupt signals to the cores. The **nIRQ** and **nFIQ** signals are controlled by software, therefore there is no requirement to tie them HIGH. If you disable the GIC CPU interface, the input pins **nVIRQ** and **nVFIQ** can be driven by an external GIC in the SoC.

If the Cortex®-A65AE core is not integrated with an external GICv3 or GICv4 distributor component in the system, then you can disable the GIC CPU interface by asserting the **GICCDISABLE** signal HIGH at reset.

GIC system register access generates **UNDEFINED** instruction exceptions when the **GICCDISABLE** signal is HIGH.

## 2.10 Advanced SIMD and floating-point support

This chapter describes the Advanced SIMD and floating-point features and registers in the Cortex®-A65AE core. The unit in charge of handling the Advanced SIMD and floating-point features is also referred to as the data engine in this manual.

### 2.10.1 About the Advanced SIMD and floating-point support

The Cortex®-A65AE core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set.

The Cortex®-A65AE floating-point implementation:

- Does not generate floating-point exceptions.
- Implements all scalar operations in hardware with support for all combinations of:
  - Rounding modes.
  - Flush-to-zero.
  - Default *Not a Number* (NaN) modes.

The Arm®v8-A architecture does not define a separate version number for its Advanced SIMD and floating-point support in the AArch64 Execution state because the instructions are always implicitly present.

## 2.10.2 Accessing the feature identification registers

Software can identify the Advanced SIMD and floating-point features using the feature identification registers in the AArch64 Execution state only.

You can access the feature identification registers in the AArch64 Execution state using the `MRS` instruction, for example:

```
MRS <Xt>, ID_AA64PFR0_EL1 ; Read ID_AA64PFR0_EL1 into Xt
```

**Table 2-27: AArch64 Advanced SIMD and scalar floating-point feature identification registers**

Register name	Description
ID_AA64ISAR0_EL1	See 3.1.53 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1 on page 152.
ID_AA64PFR0_EL1	See 3.1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1 on page 160.

## 2.11 Split-Lock feature

This chapter describes the Split-Lock feature of the Cortex®-A65AE core.

### 2.11.1 Implementing Split-Lock

The *DynamiQ Shared Unit* AE (DSU-AE) uses a specific Split-Lock implementation to enable the cluster to execute in either Split-mode, or Lock-mode, or the mixed execution Hybrid-mode. Use the **CEMODE** input to select the required cluster execution mode at boot time.

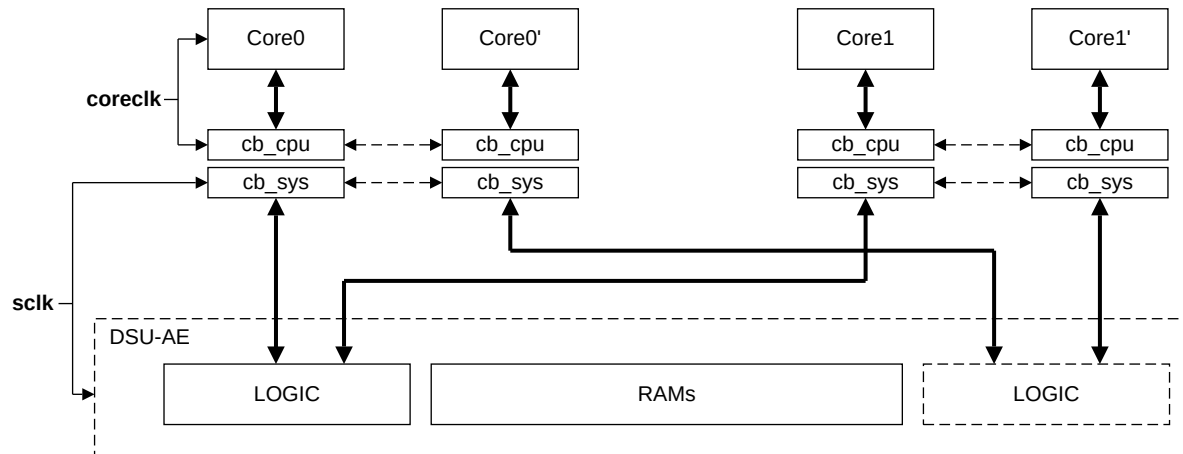
All of the DSU-AE logic, except the RAMs, is duplicated. The RAMs are shared between the two copies of the logic. RAM sharing in this way saves significant area and improves the *Failure In Time* (FIT) rate. The SECDED ECC protection scheme is always enabled for all functional DSU-AE RAMs.



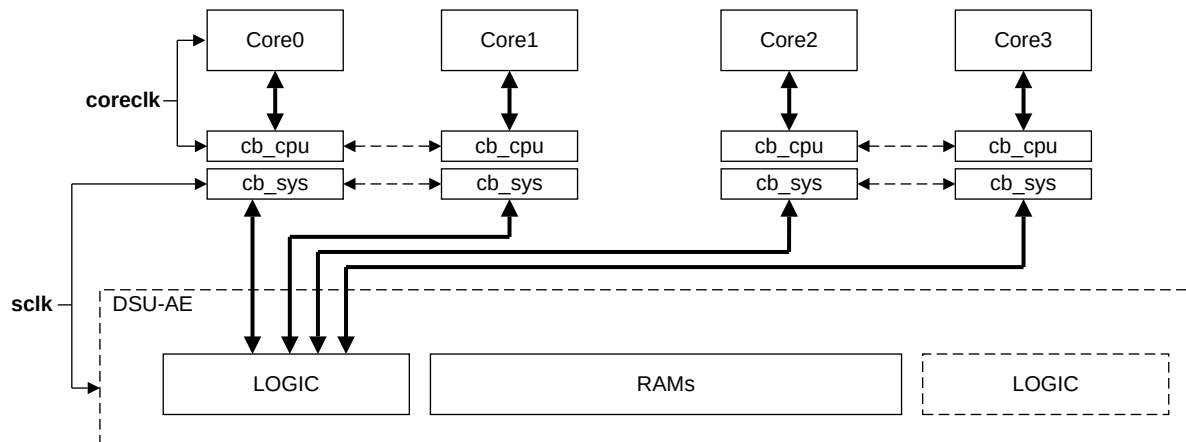
ECC protection is enabled for DSU-AE *functional* RAMs, that is, the L3 tag and data RAMs, the snoop filter, and the *Long-Term Data Buffer* (LTDB) RAM. The victim RAM is used for performance only and does not have ECC protection.

The DSU-AE uses a comparator with a registered output. In addition to the signals to be compared, the comparator includes a force input, and an enable that controls whether the compare generates an error. The force input can artificially force the comparator to generate an error result, to exercise the error reporting logic. To help protect against failures in the comparator, there are redundant copies of each of the comparators. A CPU bridge manages the asynchronous interface between the DSU-AE and the associated cores.

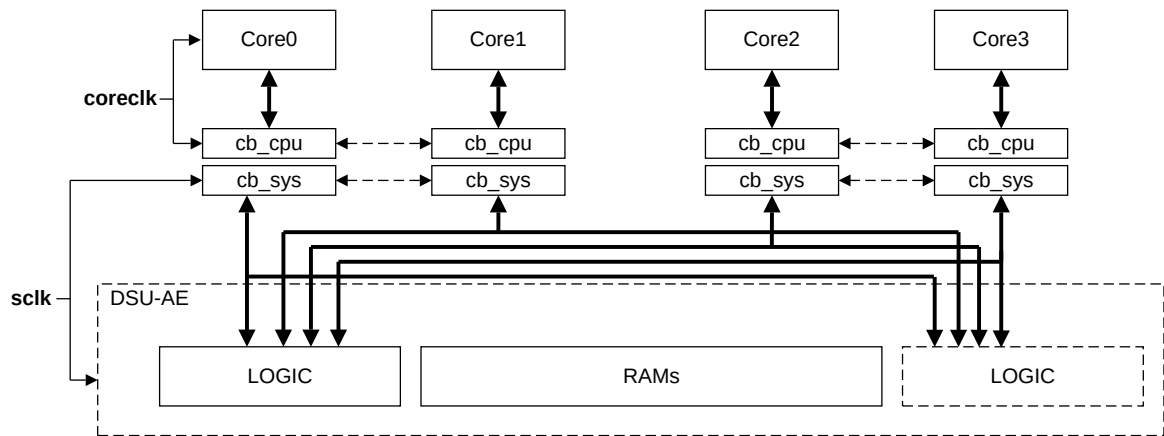
The following figure shows the DSU-AE Lock-mode operation, that is, where *CEMODE* = 0b11.

**Figure 2-8: DSU-AE Lock-mode operation**

The following figure shows the DSU-AE Split-mode operation, that is, where  $CEMODE = 0b01$ .

**Figure 2-9: DSU-AE Split-mode operation**

The following figure shows the DSU-AE Hybrid-mode operation, that is, where  $CEMODE = 0b10$ .

**Figure 2-10: DSU-AE Hybrid-mode operation**

In Split-mode, all the cores are logically present. For example, in the figure above Core0, Core1, Core 2, and Core 3 are logically present.

For more information on Split-mode, Lock-mode, and Hybrid-mode, see the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual*.

### 2.11.1.1 CPU bridge

The CPU bridge for the Cortex®-A65AE is based on the *DynamIQ Shared Unit AE* (DSU-AE) CPU bridge. The DSU-AE CPU bridge provides the asynchronous bridging functionality, and the clock and power control logic for the core.

The objectives of the CPU bridge with Split-Lock capabilities are to:

- Enable lock-step execution with temporal diversity.
- Maintain lock-step execution after crossing an asynchronous boundary.
- Increase the detection of any transient and permanent faults that affect correct program flow.
- Provide a *Fault Management Unit* (FMU) to provide fault aggregation logic and safe reporting of faults and *Reliability, Availability, and Serviceability* (RAS) core signals.
- Obtain maximum performance, due to decreased latency, when the CPU bridge is configured in synchronous mode.

The CPU bridge provides full duplication and appends each channel with logic to handle lock-step asynchronous execution.

CPU bridge synchronous mode logic is included to support CHI interfaces and contains logic to handle asynchronous powerdown requests. If a permanent or transient fault affects correct program flow, a passive mismatch occurs in the DSU-AE comparators.

The outputs from the primary CPU bridge and the redundant CPU bridge are compared to check for errors. There are two comparators for redundancy and therefore two sets of result outputs

for signaling errors. The compare outputs are qualified with valid signals whenever possible, for example for payloads. Compare outputs are also aggregated per bridge channel, for example when using CHI. Timeout detectors can detect faults where comparators alone are not sufficient. Compare fault vectors are sticky, and are controlled using core inputs for enabling and forcing mismatches in compare groups. The compare fault vectors are presented to the cluster in the system clock domain, SCLK. In addition, the CPU bridge passes safe outputs of RAS core signals in the system clock domain.

### 2.11.1.2 Comparators

Comparator logic is only enabled when the Cortex®-A65AE is operating in Lock-mode. There are two instances of each comparator, reporting on separate outputs.

Delay flops are also associated with Lock-mode.

The delay flops:

- Create the temporal diversity between the primary and redundant logic.
- Align the comparison logic.

### 2.11.1.3 Core RAS reporting signals

The CPUECTLR\_EL1[1] control bit forces reads from *Reliability, Availability, and Serviceability* (RAS) error record registers to Read-As-Zero instead of the current value in the register. This behavior prevents a read from these registers from causing divergence after a fault, where the only divergence is the updating of one or more of the RAS register bits.

Instead of register reads, the RAS register bits are output as pins from both the primary and redundant cores. In the *DynamlQ Shared Unit AE* (DSU-AE), these outputs correspond to core RAS reporting signals that the DSU-AE reports on the cluster output ports. Therefore, no divergence checks occur on these signals between the primary and redundant cores. The SoC is responsible for performing any required logical operations on these signals.

The following table shows the RAS error signals.

**Table 2-28: RAS error signals**

Core Signal	Core RAS reporting signal	Direction	Description
cpu_errmisc0[47:0]	COREERRMISC_CP<cp>_<P/R>[47:0]	Output	Current state of <b>ERR0MISC0[47:0]</b>
cpu_errstatus_ue	COREERR_UE_CP<cp>_<P/R>	Output	Current state of <b>ERR0STATUS.UE</b>
cpu_errstatus_de	COREERR_DE_CP<cp>_<P/R>	Output	Current state of <b>ERR0STATUS.DE</b>
cpu_errstatus_ce[1:0]	COREERR_CE_CP<cp>_<P/R>[1:0]	Output	Current state of <b>ERR0STATUS.CE</b>
cpu_errstatus_of	COREERR_OF_CP<cp>_<P/R>	Output	Current state of <b>ERR0STATUS.OF</b>

Core Signal	Core RAS reporting signal	Direction	Description
cpu_err_valid	COREERR_V_CP<cp>_<P/R>	Output	One-cycle pulse for each new error recorded.  <b>Note:</b> Although the preceding signals might not actually change, <b>cpu_err_valid</b> is still asserted.



These error signals are reported to the DSU-AE. For more information, see the *Core RAS reporting signals* section in the *Arm® DynamIQ™ Shared Unit AE Integration Manual*.

Each core in the Cortex®-A65AE core pair routes the error signals through the CPU bridge. The CPU bridges contain the Split-Lock functionality that monitors divergence within the core pair. The CPU bridge fault management block aggregates all faults into a single vector and reports it with redundancy to the DSU-AE, as the following figure shows.

**Figure 2-11: coredclsfault\_p/r[7:0] CPU bridge fault vector**

7	6	5	4	3	2	1	0
MISC	SRI	EVENT	TS	APB	ATB	GIC	CHI

The following table shows the CPU bridge fault vector bit assignments.

**Table 2-29: coredclsfault\_p/r[7:0] bit assignments**

Bits	Name	Function	Comparator label
[7]	MISC	Clock, power, and reset logic	SM_DCLS_<SYS/CPU><intf>_<PR/RD>
[6]	SRI	System register timer logic	SM_DCLS_<SYS/CPU>SRI_<PR/RD>
[5]	EVENT	Event logic	SM_DCLS_<SYS/CPU>EVT_<PR/RD>
[4]	TS	Timestamp logic	SM_DCLS_<SYS/CPU>TS_<PR/RD>
[3]	APB	Debug logic	SM_DCLS_<SYS/CPU>DBG_<PR/RD>
[2]	ATB	Trace logic	SM_DCLS_<SYS/CPU>ATB_<PR/RD>
[1]	GIC	Generic Interrupt Controller (GIC) logic	SM_DCLS_<SYS/CPU>GIC_<PR/RD>
[0]	CHI	CHI logic	SM_DCLS_<SYS/CPU>CHI_<PR/RD>

#### 2.11.1.4 Detectable faults

In Lock-mode, comparators can detect a single fault occurring in either the functional logic or the redundant copy of the logic. The fault is discovered when it causes a difference in the compared outputs.

A fault in either of the comparators might also be detected.

Multiple faults occurring simultaneously might also be detected when the following statements are true:



- The same fault is not present in both the functional logic and the redundant copy at the same time.
- The faults cause a difference between the compared outputs of the functional logic and those of the redundant copy.



Some faults might not cause incorrect operation. For example, a fault in a register that is never read by the software running in a particular system might not cause incorrect operation.

#### 2.11.1.5 Non-detectable faults

In Lock-mode, faults that do not cause any difference in observable behavior between the primary logic and the redundant logic are not detected. Systematic faults in the primary logic are not detected by the comparators, because the fault results in identical erroneous behavior in both primary and redundant logic.

#### 2.11.1.6 Fault containment

Errors that are detected in Lock-mode cannot be contained. The error on an output, to the external system or on one of the DSU-AE RAMs, might only be detected several cycles after it appears at the output. Therefore, it has potentially propagated into the system or into the RAM.

#### 2.11.1.7 Fault reaction

The Cortex®-A65AE processor does not include any specific features to react to a fault detected by the lock-step mechanism. The system integrator might choose to reset the system on detecting a fault or initiate some other hardware or software recovery mechanism. It is not normally possible to discover whether the fault occurred in the functional logic, in the redundant copy, or in the comparators themselves.

### Fault Reaction Time

It is not possible to quantify the *Fault Reaction Time* FRT for a fault in the processor that is detected by the lock-step mechanism. The reasons for this include:

- The fault might cause a bit in an internal register to be flipped. Until that register is read and affects the primary outputs of the processor, which might be many cycles later, the fault stays undetected. Alternatively, if this register is not used by the software running on the processor, it might never be detected.
- The number of clock cycles that are needed to propagate the fault to a primary output depends on the number of register (pipeline or buffer) stages between the fault location and the primary output.

## Latent fault detection and control mechanisms

The latent faults that could lead to non-detection of faults either within the primary or redundant core are expected to occur within the delay registers or comparator elements. These elements are defined by the system integrator. The system integrator can include latent fault detection mechanisms in the delay and comparator elements as necessary. Faults which do not affect the externally observable behavior of one of the processors in a lock-step configuration are not detectable by the lock-step comparators.

## 3 Register descriptions

This part describes the system registers of the Cortex®-A65AE core.

### 3.1 AArch64 system registers

This chapter describes the system registers in the AArch64 state.

#### 3.1.1 AArch64 registers

This chapter provides information about the AArch64 System registers with **IMPLEMENTATION DEFINED** bit fields and **IMPLEMENTATION DEFINED** registers associated with the core.

The chapter provides **IMPLEMENTATION SPECIFIC** information, for a complete description of the registers, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

As a *Simultaneously MultiThreaded* (SMT) core, the Cortex®-A65AE core supports two execution threads on each core. Each thread is a separate architectural processing element (PE), and so has a complete copy of the architectural state. Therefore each thread accesses an identical set of the architectural registers.

The chapter is presented as follows:

##### **AArch64 architectural System register summary**

This section identifies the AArch64 architectural System registers implemented in the Cortex®-A65AE core that have **IMPLEMENTATION DEFINED** bit fields. The register descriptions for these registers only contain information about the **IMPLEMENTATION DEFINED** bits.

##### **AArch64 IMPLEMENTATION DEFINED register summary**

This section identifies the AArch64 architectural registers that are implemented in the Cortex®-A65AE core that are **IMPLEMENTATION DEFINED**.

##### **AArch64 registers by functional group**

This section groups the **IMPLEMENTATION DEFINED** registers and architectural System registers with **IMPLEMENTATION DEFINED** bit fields, as identified previously, by function. It also provides reset details for key register types.

##### **Register descriptions**

The remainder of the chapter provides register descriptions of the **IMPLEMENTATION DEFINED** registers and architectural system registers with **IMPLEMENTATION DEFINED** bit fields, as identified previously. These are listed in alphabetic order.

### 3.1.2 AArch64 architectural system register summary

This section describes the AArch64 architectural system registers implemented in the Cortex®-A65AE core.

The section contains two tables:

#### Registers with implementation defined bit fields

This table identifies the architecturally defined registers in Cortex®-A65AE that have implementation defined bit fields. The register descriptions for these registers only contain information about the implementation defined bits.

See [Registers with implementation defined bit fields](#) on page 92.

#### Other architecturally defined registers

This table identifies the other architecturally defined registers that are implemented in the Cortex®-A65AE core. These registers are described in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

See [Other architecturally defined registers](#) on page 94.

**Table 3-1: Registers with implementation defined bit fields**

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
ACTLR_EL1	3	c1	0	c0	1	64	<a href="#">3.1.5 ACTLR_EL1, Auxiliary Control Register, EL1</a> on page 101
ACTLR_EL2	3	c1	4	c0	1	64	<a href="#">3.1.6 ACTLR_EL2, Auxiliary Control Register, EL2</a> on page 102
ACTLR_EL3	3	c1	6	c0	1	64	<a href="#">3.1.7 ACTLR_EL3, Auxiliary Control Register, EL3</a> on page 104
AFSR0_EL1	3	c5	0	c1	0	32	<a href="#">3.1.8 AFSR0_EL1, Auxiliary Fault Status Register 0, EL1</a> on page 106
AFSR0_EL2	3	c5	4	c1	0	32	<a href="#">3.1.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2</a> on page 106
AFSR0_EL3	3	c5	6	c1	0	32	<a href="#">3.1.10 AFSR0_EL3, Auxiliary Fault Status Register 0, EL3</a> on page 107
AFSR1_EL1	3	c5	0	c1	1	32	<a href="#">3.1.11 AFSR1_EL1, Auxiliary Fault Status Register 1, EL1</a> on page 107
AFSR1_EL2	3	c5	4	c1	1	32	<a href="#">3.1.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2</a> on page 107
AFSR1_EL3	3	c5	6	c1	1	32	<a href="#">3.1.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3</a> on page 107
AIDR_EL1	3	c0	1	c0	7	32	<a href="#">3.1.14 AIDR_EL1, Auxiliary ID Register, EL1</a> on page 107
AMAIR_EL1	3	c10	0	c3	0	64	<a href="#">3.1.15 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register, EL1</a> on page 108
AMAIR_EL2	3	c10	4	c3	0	64	<a href="#">3.1.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2</a> on page 108
AMAIR_EL3	3	c10	6	c3	0	64	<a href="#">3.1.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3</a> on page 108
CCSIDR_EL1	3	c0	1	c0	0	32	<a href="#">3.1.18 CCSIDR_EL1, Cache Size ID Register, EL1</a> on page 108
CLIDR_EL1	3	c0	1	c0	1	64	<a href="#">3.1.19 CLIDR_EL1, Cache Level ID Register, EL1</a> on page 110
CPACR_EL1	3	c1	0	c0	2	32	<a href="#">3.1.20 CPACR_EL1, Architectural Feature Access Control Register, EL1</a> on page 112
CPTR_EL2	3	c1	4	c1	2	32	<a href="#">3.1.21 CPTL_EL2, Architectural Feature Trap Register, EL2</a> on page 113
CPTR_EL3	3	c1	6	c1	2	32	<a href="#">3.1.22 CPTL_EL3, Architectural Feature Trap Register, EL3</a> on page 117
CSSELR_EL1	3	c0	2	c0	0	32	<a href="#">3.1.31 CSSELR_EL1, Cache Size Selection Register, EL1</a> on page 133

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
CTR_ELO	3	c0	3	c0	1	32	3.1.32 CTR_ELO, Cache Type Register, ELO on page 134
DISR_EL1	3	c12	0	c1	1	64	3.1.34 DISR_EL1, Deferred Interrupt Status Register, EL1 on page 137
ERRIDR_EL1	3	c5	0	c3	0	32	3.1.35 ERRIDR_EL1, Error ID Register, EL1 on page 138
ERRSELR_EL1	3	c5	0	c3	1	32	3.1.36 ERRSELR_EL1, Error Record Select Register, EL1 on page 139
ERXCTLR_EL1	3	c5	0	c4	1	64	3.1.37 ERXCTLR_EL1, Selected Error Record Control Register, EL1 on page 140
ERXFR_EL1	3	c5	0	c4	0	64	3.1.38 ERXFR_EL1, Selected Error Record Feature Register, EL1 on page 140
ERXMISCO_EL1	3	c5	0	c5	0	64	3.1.39 ERXMISCO_EL1, Selected Error Record Miscellaneous Register 0, EL1 on page 141
ERXSTATUS_EL1	3	c5	0	c4	2	32	3.1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1 on page 145
ESR_EL1	3	c5	0	c2	0	32	3.1.44 ESR_EL1, Exception Syndrome Register, EL1 on page 145
ESR_EL2	3	c5	4	c2	0	32	3.1.45 ESR_EL2, Exception Syndrome Register, EL2 on page 146
ESR_EL3	3	c5	6	c2	0	32	3.1.46 ESR_EL3, Exception Syndrome Register, EL3 on page 147
HACR_EL2	3	c1	4	c1	7	32	3.1.47 HACR_EL2, Hyp Auxiliary Configuration Register, EL2 on page 148
HCR_EL2	3	c1	4	c1	0	64	3.1.48 HCR_EL2, Hypervisor Configuration Register, EL2 on page 148
ID_AA64DFR0_EL1	3	c0	0	c5	0	64	3.1.51 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0, EL1 on page 150
ID_AA64ISAR0_EL1	3	c0	0	c6	0	64	3.1.53 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1 on page 152
ID_AA64ISAR1_EL1	3	c0	0	c6	1	64	3.1.54 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1 on page 154
ID_AA64MMFR0_EL1	3	c0	0	c7	0	64	3.1.55 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0, EL1 on page 155
ID_AA64MMFR1_EL1	3	c0	0	c7	1	64	3.1.56 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1 on page 157
ID_AA64MMFR2_EL1	3	c0	0	c7	2	64	3.1.57 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1 on page 159
ID_AA64PFR0_EL1	3	c0	0	c4	0	64	3.1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1 on page 160
LORC_EL1	3	c10	0	c4	3	64	3.1.60 LORC_EL1, LORegion Control Register, EL1 on page 164
LORID_EL1	3	c10	0	c4	7	64	3.1.61 LORID_EL1, LORegion ID Register, EL1 on page 164
LORN_EL1	3	c10	0	c4	2	64	3.1.62 LORN_EL1, LORegion Number Register, EL1 on page 165
MDCR_EL3	3	c1	6	c3	1	32	3.1.63 MDCR_EL3, Monitor Debug Configuration Register, EL3 on page 166
MIDR_EL1	3	c0	0	c0	0	32	3.1.64 MIDR_EL1, Main ID Register, EL1 on page 168
MPIDR_EL1	3	c0	0	c0	5	64	3.1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1 on page 169
PAR_EL1	3	c7	0	c4	0	64	3.1.66 PAR_EL1, Physical Address Register, EL1 on page 171
REVIDR_EL1	3	c0	0	c0	6	32	3.1.67 REVIDR_EL1, Revision ID Register, EL1 on page 172
RVBAR_EL3	3	c12	6	c0	1	64	3.1.69 RVBAR_EL3, Reset Vector Base Address Register, EL3 on page 173
SCTLR_EL1	3	c1	0	c0	0	32	3.1.70 SCTLR_EL1, System Control Register, EL1 on page 174
SCTLR_EL2	3	c1	4	c0	0	32	3.1.71 SCTLR_EL2, System Control Register, EL2 on page 175

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
SCTLR_EL12	3	c1	5	c0	0	32	<a href="#">3.1.70 SCTLR_EL1, System Control Register, EL1</a> on page 174
SCTLR_EL3	3	c1	6	c0	0	32	<a href="#">3.1.72 SCTLR_EL3, System Control Register, EL3</a> on page 176
TCR_EL1	3	c2	0	c0	2	64	<a href="#">3.1.73 TCR_EL1, Translation Control Register, EL1</a> on page 178
TCR_EL2	3	c2	4	c0	2	64	<a href="#">3.1.74 TCR_EL2, Translation Control Register, EL2</a> on page 179
TCR_EL3	3	c2	6	c0	2	64	<a href="#">3.1.75 TCR_EL3, Translation Control Register, EL3</a> on page 180
TTBR0_EL1	3	c2	0	c0	0	64	<a href="#">3.1.76 TTBR0_EL1, Translation Table Base Register 0, EL1</a> on page 182
TTBR0_EL2	3	c2	4	c0	0	64	<a href="#">3.1.77 TTBR0_EL2, Translation Table Base Register 0, EL2</a> on page 183
TTBR0_EL3	3	c2	6	c0	0	64	<a href="#">3.1.78 TTBR0_EL3, Translation Table Base Register 0, EL3</a> on page 185
TTBR1_EL1	3	c2	0	c0	1	64	<a href="#">3.1.79 TTBR1_EL1, Translation Table Base Register 1, EL1</a> on page 186
TTBR1_EL2	3	c2	4	c0	1	64	<a href="#">3.1.80 TTBR1_EL2, Translation Table Base Register 1, EL2</a> on page 187
VDISR_EL2	3	c12	4	c1	1	64	<a href="#">3.1.81 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2</a> on page 187
VSESR_EL2	3	c5	4	c2	3	64	<a href="#">3.1.82 VSESR_EL2, Virtual SError Exception Syndrome Register</a> on page 188
VTCCR_EL2	3	c2	4	c1	2	32	<a href="#">3.1.83 VTCCR_EL2, Virtualization Translation Control Register, EL2</a> on page 189
VTTBR_EL2	3	c2	4	c1	0	64	<a href="#">3.1.84 VTTBR_EL2, Virtualization Translation Table Base Register, EL2</a> on page 190

**Table 3-2: Other architecturally defined registers**

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
AFSR0_EL12	3	c5	5	1	0	32	Auxiliary Fault Status Register 0
AFSR1_EL12	3	c5	5	1	1	32	Auxiliary Fault Status Register 1
AMAIR_EL12	3	c10	5	c3	0	64	Auxiliary Memory Attribute Indirection Register
CNTFRQ_ELO	3	c14	3	0	0	32	Counter-timer Frequency register
CNTHCTL_EL2	3	c14	4	c1	0	32	Counter-timer Hypervisor Control register
CNTHP_CTL_EL2	3	c14	4	c2	1	32	Counter-timer Hypervisor Physical Timer Control register
CNTHP_CVAL_EL2	3	c14	4	c2	2	64	Counter-timer Hyp Physical CompareValue register
CNTHP_TVAL_EL2	3	c14	4	c2	0	32	Counter-timer Hyp Physical Timer TimerValue register
CNTHV_CTL_EL2	3	c14	4	c3	1	32	Counter-timer Virtual Timer Control register
CNTHV_CVAL_EL2	3	c14	4	c3	2	64	Counter-timer Virtual Timer CompareValue register
CNTHV_TVAL_EL2	3	c14	4	c3	0	32	Counter-timer Virtual Timer TimerValue register
CNTKCTL_EL1	3	c14	0	c1	0	32	Counter-timer Kernel Control register
CNTKCTL_EL12	3	c14	5	c1	0	32	Counter-timer Kernel Control register
CNTP_CTL_ELO	3	c14	3	c2	1	32	Counter-timer Physical Timer Control register
CNTP_CTL_EL02	3	c14	5	c2	1	32	Counter-timer Physical Timer Control register
CNTP_CVAL_ELO	3	c14	3	c2	2	64	Counter-timer Physical Timer CompareValue register
CNTP_CVAL_EL02	3	c14	5	c2	2	64	Counter-timer Physical Timer CompareValue register
CNTP_TVAL_ELO	3	c14	3	c2	0	32	Counter-timer Physical Timer TimerValue register
CNTP_TVAL_EL02	3	c14	5	c2	0	32	Counter-timer Physical Timer TimerValue register
CNTPCT_ELO	3	c14	3	c0	1	64	Counter-timer Physical Count register
CNTPS_CTL_EL1	3	c14	7	c2	1	32	Counter-timer Physical Secure Timer Control register

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
CNTPS_CVAL_EL1	3	c14	7	c2	2	64	Counter-timer Physical Secure Timer CompareValue register
CNTPS_TVAL_EL1	3	c14	7	c2	0	32	Counter-timer Physical Secure Timer TimerValue register
CNTV_CTL_EL0	3	c14	3	c3	1	32	Counter-timer Virtual Timer Control register
CNTV_CTL_EL02	3	c14	5	c3	1	32	Counter-timer Virtual Timer Control register
CNTV_CVAL_EL0	3	c14	3	c3	2	64	Counter-timer Virtual Timer CompareValue register
CNTV_CVAL_EL02	3	c14	5	c3	2	64	Counter-timer Virtual Timer CompareValue register
CNTV_TVAL_EL0	3	c14	3	c3	0	32	Counter-timer Virtual Timer TimerValue register
CNTV_TVAL_EL02	3	c14	5	c3	0	32	Counter-timer Virtual Timer TimerValue register
CNTVCT_EL0	3	c14	3	c0	2	64	Counter-timer Virtual Count register
CNTVOFF_EL2	3	c14	4	c0	3	64	Counter-timer Virtual Offset register
CONTEXTIDR_EL1	3	c13	0	c0	1	32	Context ID Register (EL1)
CONTEXTIDR_EL12	3	c13	5	c0	1	32	Context ID Register (EL12)
CONTEXTIDR_EL2	3	c13	4	c0	1	32	Context ID Register (EL2)
CPACR_EL12	3	c1	5	c0	2	32	Architectural Feature Access Control Register
CPTR_EL3	3	c1	6	c1	2	32	Architectural Feature Trap Register (EL3)
DACR32_EL2	3	c3	4	c0	0	32	Domain Access Control Register
ESR_EL12	3	c5	5	c2	0	32	Exception Syndrome Register (EL12)
FAR_EL1	3	c6	0	c0	0	64	Fault Address Register (EL1)
FAR_EL12	3	c6	5	c0	0	64	Fault Address Register (EL12)
FAR_EL2	3	c6	4	c0	0	64	Fault Address Register (EL2)
FAR_EL3	3	c6	6	c0	0	64	Fault Address Register (EL3)
FPEXC32_EL2	3	c5	4	c3	0	32	Floating-point Exception Control register
HPFAR_EL2	3	c6	4	c0	4	64	Hypervisor IPA Fault Address Register
HSTR_EL2	3	c1	4	c1	3	32	Hypervisor System Trap Register
ID_AA64AFR0_EL1	3	c0	0	c5	4	64	AArch64 Auxiliary Feature Register 0
ID_AA64AFR1_EL1	3	c0	0	c5	5	64	AArch64 Auxiliary Feature Register 1
ID_AA64DFR1_EL1	3	c0	0	c5	1	64	AArch64 Debug Feature Register 1
ID_AA64PFR1_EL1	3	c0	0	c4	1	64	AArch64 Core Feature Register 1
ISR_EL1	3	c12	0	c1	0	32	Interrupt Status Register
LOREA_EL1	3	c10	0	c4	1	64	LORegion End Address Register
LORSA_EL1	3	c10	0	c4	0	64	LORegion Start Address Register
MAIR_EL1	3	c10	0	c2	0	64	Memory Attribute Indirection Register (EL1)
MAIR_EL12	3	c10	5	c2	0	64	Memory Attribute Indirection Register (EL12)
MAIR_EL2	3	c10	4	c2	0	64	Memory Attribute Indirection Register (EL2)
MAIR_EL3	3	c10	6	c2	0	64	Memory Attribute Indirection Register (EL3)
MDCR_EL2	3	c1	4	c1	1	32	Monitor Debug Configuration Register
MVFR0_EL1	3	c0	0	c3	0	32	AArch32 Media and VFP Feature Register 0
MVFR1_EL1	3	c0	0	c3	1	32	AArch32 Media and VFP Feature Register 1
MVFR2_EL1	3	c0	0	c3	2	32	AArch32 Media and VFP Feature Register 2
RMR_EL3	3	c12	6	c0	2	32	Reset Management Register

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
SCR_EL3	3	c1	6	c1	0	32	Secure Configuration Register
SDER32_EL3	3	c1	6	c1	1	32	AArch32 Secure Debug Enable Register
TCR_EL12	3	c2	5	c0	2	64	Translation Control Register (EL12)
TPIDR_ELO	3	c13	3	c0	2	64	ELO Read/Write Software Thread ID Register
TPIDR_EL1	3	c13	0	c0	4	64	EL1 Software Thread ID Register
TPIDR_EL2	3	c13	4	c0	2	64	EL2 Software Thread ID Register
TPIDR_EL3	3	c13	6	c0	2	64	EL3 Software Thread ID Register
TPIDRRO_ELO	3	c13	3	c0	3	64	ELO Read-Only Software Thread ID Register
TTBR0_EL12	3	c2	5	c0	0	64	Translation Table Base Register 0 (EL12)
TTBR1_EL12	3	c2	5	c0	1	64	Translation Table Base Register 1 (EL12)
VBAR_EL1	3	c12	0	c0	0	64	Vector Base Address Register (EL1)
VBAR_EL12	3	c12	5	c0	0	64	Vector Base Address Register (EL12)
VBAR_EL2	3	c12	4	c0	0	64	Vector Base Address Register (EL2)
VBAR_EL3	3	c12	6	c0	0	64	Vector Base Address Register (EL3)
VMPIDR_EL2	3	c0	4	c0	5	64	Virtualization Multiprocessor ID Register
VPIDR_EL2	3	c0	4	c0	0	32	Virtualization Core ID Register

### 3.1.3 AArch64 IMPLEMENTATION DEFINED register summary

This section describes the AArch64 registers in the Cortex®-A65AE core that are **IMPLEMENTATION DEFINED**.

The following tables lists the AArch64 **IMPLEMENTATION DEFINED** registers, sorted by opcode.

**Table 3-3: AArch64 IMPLEMENTATION DEFINED registers**

Name	Copro	CRn	Op1	CRm	Op2	Width	Description
CPUACTLR_EL1	3	c15	0	c1	0	64	<a href="#">3.1.23 CPUACTLR_EL1, CPU Auxiliary Control Register, EL1</a> on page 118
CPUCFR_EL1	3	c15	0	c0	0	32	<a href="#">3.1.24 CPUCFR_EL1, CPU Configuration Register, EL1</a> on page 119
CPUECTLR_EL1	3	c15	0	c1	4	64	<a href="#">3.1.25 CPUECTLR_EL1, CPU Extended Control Register, EL1</a> on page 121
CPUPCR_EL3	3	15	6	c8	1	64	<a href="#">3.1.26 CPUPCR_EL3, CPU Private Control Register, EL3</a> on page 125
CPUPMR_EL3	3	c15	6	c8	3	64	<a href="#">3.1.27 CPUPMR_EL3, CPU Private Mask Register, EL3</a> on page 126
CPUPOR_EL3	3	c15	6	c8	2	64	<a href="#">3.1.28 CPUPOR_EL3, CPU Private Operation Register, EL3</a> on page 128
CPUPSELR_EL3	3	c15	6	c8	0	32	<a href="#">3.1.29 CPUPSELR_EL3, CPU Private Selection Register, EL3</a> on page 129
CPUPWRCTLR_EL1	3	c15	0	c2	7	32	<a href="#">3.1.30 CPUPWRCTLR_EL1, Power Control Register, EL1</a> on page 131
ERXPFPGCDN_EL1	3	c15	0	c2	2	32	<a href="#">3.1.40 ERXPFPGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1</a> on page 141
ERXPFGCTL_EL1	3	c15	0	c2	1	32	<a href="#">3.1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1</a> on page 142
ERXPFGF_EL1	3	c15	0	c2	0	32	<a href="#">3.1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1</a> on page 144



### 3.1.4 AArch64 registers by functional group

This section identifies the AArch64 registers by their functional groups and applies to the registers in the core that are **IMPLEMENTATION DEFINED** or have micro-architectural bit fields. Reset values are provided for these registers.

**Table 3-4: Identification registers**

Name	Type	Reset	Description
CCSIDR_EL1	RO	-	3.1.18 CCSIDR_EL1, Cache Size ID Register, EL1 on page 108
CLIDR_EL1	RO	<ul style="list-style-type: none"> <li>0xC3000123 if L3 cache present.</li> <li>0x82000023 if no L3 cache.</li> </ul>	3.1.19 CLIDR_EL1, Cache Level ID Register, EL1 on page 110
CSSELR_EL1	RW	UNK	3.1.31 CSSELR_EL1, Cache Size Selection Register, EL1 on page 133
CTR_ELO	RO	0x84448004	3.1.32 CTR_ELO, Cache Type Register, ELO on page 134
DCZID_ELO	RO	0x00000004	3.1.33 DCZID_ELO, Data Cache Zero ID Register, ELO on page 136
ERRIDR_EL1	RO	-	3.1.35 ERRIDR_EL1, Error ID Register, EL1 on page 138
ID_AA64AFR0_EL1	RO	0x00000000	3.1.49 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0 on page 150
ID_AA64AFR1_EL1	RO	0x00000000	3.1.50 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1 on page 150
ID_AA64DFR0_EL1	RO	0x0000000010305408	3.1.51 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0, EL1 on page 150
ID_AA64DFR1_EL1	RO	0x00000000	3.1.52 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1, EL1 on page 152
ID_AA64ISAR0_EL1	RO	<ul style="list-style-type: none"> <li>0x0000100010211120 if the Cryptographic Extension is implemented.</li> <li>0x0000100010210000 if the Cryptographic Extension is not implemented.</li> </ul>	3.1.53 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1 on page 152
ID_AA64ISAR1_EL1	RO	0x0000000000100001	3.1.54 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1 on page 154
ID_AA64MMFR0_EL1	RO	0x0000000000101124	3.1.55 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0, EL1 on page 155
ID_AA64MMFR1_EL1	RO	0x0000000010212122	3.1.56 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1 on page 157
ID_AA64MMFR2_EL1	RO	0x00000000000001011	3.1.57 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1 on page 159
ID_AA64PFR0_EL1	RO	<ul style="list-style-type: none"> <li>0x0000000010112222 if the GICv4 interface is disabled.</li> <li>0x0000000011112222 if the GICv4 interface is enabled.</li> </ul>	3.1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1 on page 160

Name	Type	Reset	Description
ID_AA64PFR1_EL1	RO	0x0000000000000010	3.1.59 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1, EL1 on page 163
LORID_EL1	RO	0x0000000000004004	3.1.61 LORID_EL1, LORegion ID Register, EL1 on page 164
MIDR_EL1	RO	0x411FD430	3.1.64 MIDR_EL1, Main ID Register, EL1 on page 168
MPIDR_EL1	RO	The reset value depends on <b>CLUSTERIDAFF2[7:0]</b> and <b>CLUSTERIDAFF3[7:0]</b> . See register description for details.	3.1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1 on page 169
REVIDR_EL1	RO	0x00000000	3.1.67 REVIDR_EL1, Revision ID Register, EL1 on page 172
VMPIDR_EL2	RW	The reset value is the value of MPIDR_EL1.	Virtualization Multiprocessor ID Register EL2
VPIDR_EL2	RW	The reset value is the value of MIDR_EL1.	Virtualization Core ID Register EL2

**Table 3-5: Other system control registers**

Name	Type	Description
ACTLR_EL1	RW	3.1.5 ACTLR_EL1, Auxiliary Control Register, EL1 on page 101
ACTLR_EL2	RW	3.1.6 ACTLR_EL2, Auxiliary Control Register, EL2 on page 102
ACTLR_EL3	RW	3.1.7 ACTLR_EL3, Auxiliary Control Register, EL3 on page 104
CPACR_EL1	RW	3.1.20 CPACR_EL1, Architectural Feature Access Control Register, EL1 on page 112
SCTLR_EL1	RW	3.1.70 SCTLR_EL1, System Control Register, EL1 on page 174
SCTLR_EL2	RW	3.1.71 SCTLR_EL2, System Control Register, EL2 on page 175
SCTLR_EL3	RW	3.1.72 SCTLR_EL3, System Control Register, EL3 on page 176
SCTLR_EL12	RW	3.1.70 SCTLR_EL1, System Control Register, EL1 on page 174

**Table 3-6: Reliability, Availability, Serviceability (RAS) registers**

Name	Type	Description
DISR_EL1	RW	3.1.34 DISR_EL1, Deferred Interrupt Status Register, EL1 on page 137
ERRIDR_EL1	RW	3.1.35 ERRIDR_EL1, Error ID Register, EL1 on page 138
ERRSELR_EL1	RW	3.1.36 ERRSELR_EL1, Error Record Select Register, EL1 on page 139
ERXCTLR_EL1	RW	3.1.37 ERXCTLR_EL1, Selected Error Record Control Register, EL1 on page 140
ERXFR_EL1	RO	3.1.38 ERXFR_EL1, Selected Error Record Feature Register, EL1 on page 140
ERXMISCO_EL1	RW	3.1.39 ERXMISCO_EL1, Selected Error Record Miscellaneous Register 0, EL1 on page 141
ERXSTATUS_EL1	RW	3.1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1 on page 145
ERXPFGCDN_EL1	RW	3.1.40 ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1 on page 141
ERXPFGCTL_EL1	RW	3.1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1 on page 142
ERXPFGF_EL1	RO	3.1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1 on page 144
HCR_EL2	RW	3.1.48 HCR_EL2, Hypervisor Configuration Register, EL2 on page 148
VDISR_EL2	RW	3.1.81 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2 on page 187
VSESR_EL2	RW	3.1.82 VSESR_EL2, Virtual SError Exception Syndrome Register on page 188

**Table 3-7: Virtual Memory control registers**

Name	Type	Description
AMAIR_EL1	RW	<a href="#">3.1.15 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register, EL1</a> on page 108
AMAIR_EL2	RW	<a href="#">3.1.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2</a> on page 108
AMAIR_EL3	RW	<a href="#">3.1.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3</a> on page 108
LORC_EL1	RW	<a href="#">3.1.60 LORC_EL1, LORegion Control Register, EL1</a> on page 164
LOREA_EL1	RW	LORegion End Address Register EL1
LORID_EL1	RO	<a href="#">3.1.61 LORID_EL1, LORegion ID Register, EL1</a> on page 164
LORN_EL1	RW	<a href="#">3.1.62 LORN_EL1, LORegion Number Register, EL1</a> on page 165
LORSA_EL1	RW	LORegion Start Address Register EL1
TCR_EL1	RW	<a href="#">3.1.73 TCR_EL1, Translation Control Register, EL1</a> on page 178
TCR_EL2	RW	<a href="#">3.1.74 TCR_EL2, Translation Control Register, EL2</a> on page 179
TCR_EL3	RW	<a href="#">3.1.75 TCR_EL3, Translation Control Register, EL3</a> on page 180
TTBR0_EL1	RW	<a href="#">3.1.76 TTBR0_EL1, Translation Table Base Register 0, EL1</a> on page 182
TTBR0_EL2	RW	<a href="#">3.1.77 TTBR0_EL2, Translation Table Base Register 0, EL2</a> on page 183
TTBR0_EL3	RW	<a href="#">3.1.78 TTBR0_EL3, Translation Table Base Register 0, EL3</a> on page 185
TTBR1_EL1	RW	<a href="#">3.1.79 TTBR1_EL1, Translation Table Base Register 1, EL1</a> on page 186
TTBR1_EL2	RW	<a href="#">3.1.80 TTBR1_EL2, Translation Table Base Register 1, EL2</a> on page 187
VTTBR_EL2	RW	<a href="#">3.1.84 VTTBR_EL2, Virtualization Translation Table Base Register, EL2</a> on page 190

**Table 3-8: Virtualization registers**

Name	Type	Description
ACTLR_EL2	RW	<a href="#">3.1.6 ACTLR_EL2, Auxiliary Control Register, EL2</a> on page 102
AFSR0_EL2	RW	<a href="#">3.1.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2</a> on page 106
AFSR1_EL2	RW	<a href="#">3.1.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2</a> on page 107
AMAIR_EL2	RW	<a href="#">3.1.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2</a> on page 108
CPTR_EL2	RW	<a href="#">3.1.21 CPTR_EL2, Architectural Feature Trap Register, EL2</a> on page 113
ESR_EL2	RW	<a href="#">3.1.45 ESR_EL2, Exception Syndrome Register, EL2</a> on page 146
HACR_EL2	RW	<a href="#">3.1.47 HACR_EL2, Hyp Auxiliary Configuration Register, EL2</a> on page 148
HCR_EL2	RW	<a href="#">3.1.48 HCR_EL2, Hypervisor Configuration Register, EL2</a> on page 148
HPFAR_EL2	RW	Hypervisor IPA Fault Address Register EL2
TCR_EL2	RW	<a href="#">3.1.74 TCR_EL2, Translation Control Register, EL2</a> on page 179
VMPIDR_EL2	RW	Virtualization Multiprocessor ID Register EL2
VPIDR_EL2	RW	Virtualization Core ID Register EL2
VSESR_EL2	RW	<a href="#">3.1.82 VSESR_EL2, Virtual SError Exception Syndrome Register</a> on page 188
VTCTR_EL2	RW	<a href="#">3.1.83 VTCTR_EL2, Virtualization Translation Control Register, EL2</a> on page 189
VTTBR_EL2	RW	<a href="#">3.1.84 VTTBR_EL2, Virtualization Translation Table Base Register, EL2</a> on page 190

**Table 3-9: Exception and fault handling registers**

Name	Type	Description
AFSR0_EL1	RW	<a href="#">3.1.8 AFSR0_EL1, Auxiliary Fault Status Register 0, EL1</a> on page 106

Name	Type	Description
AFSRO_EL2	RW	<a href="#">3.1.9 AFSRO_EL2, Auxiliary Fault Status Register 0, EL2</a> on page 106
AFSRO_EL3	RW	<a href="#">3.1.10 AFSRO_EL3, Auxiliary Fault Status Register 0, EL3</a> on page 107
AFSR1_EL1	RW	<a href="#">3.1.11 AFSR1_EL1, Auxiliary Fault Status Register 1, EL1</a> on page 107
AFSR1_EL2	RW	<a href="#">3.1.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2</a> on page 107
AFSR1_EL3	RW	<a href="#">3.1.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3</a> on page 107
DISR_EL1	RW	<a href="#">3.1.34 DISR_EL1, Deferred Interrupt Status Register, EL1</a> on page 137
ESR_EL1	RW	<a href="#">3.1.44 ESR_EL1, Exception Syndrome Register, EL1</a> on page 145
ESR_EL2	RW	<a href="#">3.1.45 ESR_EL2, Exception Syndrome Register, EL2</a> on page 146
ESR_EL3	RW	<a href="#">3.1.46 ESR_EL3, Exception Syndrome Register, EL3</a> on page 147
HPFAR_EL2	RW	Hypervisor IPA Fault Address Register EL2
VDISR_EL2	RW	<a href="#">3.1.81 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2</a> on page 187
VSESR_EL2	RW	<a href="#">3.1.82 VSESR_EL2, Virtual SError Exception Syndrome Register</a> on page 188

**Table 3-10: IMPLEMENTATION DEFINED registers**

Name	Type	Description
CPUACTLR_EL1	RW	<a href="#">3.1.23 CPUACTLR_EL1, CPU Auxiliary Control Register, EL1</a> on page 118
CPUCFR_EL1	RO	<a href="#">3.1.24 CPUCFR_EL1, CPU Configuration Register, EL1</a> on page 119
CPUECTLR_EL1	RW	<a href="#">3.1.25 CPUECTLR_EL1, CPU Extended Control Register, EL1</a> on page 121
CPUPWRCTLR_EL1	RW	<a href="#">3.1.30 CPUPWRCTLR_EL1, Power Control Register, EL1</a> on page 131
ERXPFPGCDN_EL1	RW	<a href="#">3.1.40 ERXPFPGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1</a> on page 141
ERXPFPGCTL_EL1	RW	<a href="#">3.1.41 ERXPFPGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1</a> on page 142
ERXPFPGF_EL1	RW	<a href="#">3.1.42 ERXPFPGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1</a> on page 144

**Table 3-11: Security**

Name	Type	Description
ACTLR_EL3	RW	<a href="#">3.1.7 ACTLR_EL3, Auxiliary Control Register, EL3</a> on page 104
AFSRO_EL3	RW	<a href="#">3.1.10 AFSRO_EL3, Auxiliary Fault Status Register 0, EL3</a> on page 107
AFSR1_EL3	RW	<a href="#">3.1.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3</a> on page 107
AMAIR_EL3	RW	<a href="#">3.1.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3</a> on page 108
CPTR_EL3	RW	<a href="#">3.1.22 CPTR_EL3, Architectural Feature Trap Register, EL3</a> on page 117
MDCR_EL3	RW	<a href="#">3.1.63 MDCR_EL3, Monitor Debug Configuration Register, EL3</a> on page 166

**Table 3-12: Reset management registers**

Name	Type	Description
RMR_EL3	RW	<a href="#">3.1.68 RMR_EL3, Reset Management Register</a> on page 172
RVBAR_EL3	RW	<a href="#">3.1.69 RVBAR_EL3, Reset Vector Base Address Register, EL3</a> on page 173

**Table 3-13: Address registers**

Name	Type	Description
PAR_EL1	RW	<a href="#">3.1.66 PAR_EL1, Physical Address Register, EL1</a> on page 171

### 3.1.5 ACTLR\_EL1, Auxiliary Control Register, EL1

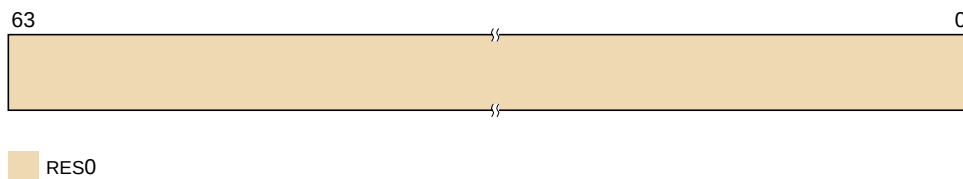
ACTLR\_EL1 provides **IMPLEMENTATION DEFINED** configuration and control options for execution at EL1 and EL0.

#### Bit field descriptions

ACTLR\_EL1 is a 64-bit register, and is part of:

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

**Figure 3-1: ACTLR\_EL1 bit assignments**



#### RES0, [63:0]

RES0      Reserved.

#### Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

#### Accessibility

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

Subject to these prioritization rules, this register provides the following access permissions.

Execution at EL3:

- Write-Access to CPUACTLR\_EL1 always.
- Write-Access to CPUECTLR\_EL1 always.

Execution at EL2:

- Write-Access to CPUACTLR\_EL1 if ACTLR\_EL3[0] = 1.
- Write-Access to CPUECTLR\_EL1 if ACTLR\_EL3[1] = 1.

Execution at EL1:

- Write-Access to CPUACTLR\_EL1 if ACTLR\_EL3[0] = 1 & (ACTLR\_EL2[0] = 1 || SCR\_EL3.NS = 0).
- Write-Access to CPEACTLR\_EL1 if ACTLR\_EL3[1] = 1 & (ACTLR\_EL2[1] = 1 || SCR\_EL3.NS = 0).

If Write-Access is not possible the instruction traps to the lowest Exception level that denied access. For example, if the ACTLR\_EL2 denies access to lower Exception levels then an attempt at EL1 traps to EL2.

### 3.1.6 ACTLR\_EL2, Auxiliary Control Register, EL2

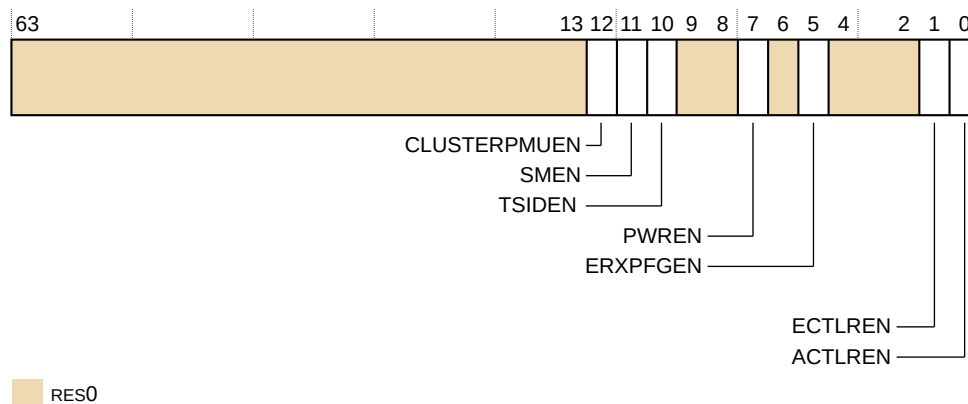
The ACTLR\_EL2 provides **IMPLEMENTATION DEFINED** configuration and control options for EL2.

#### Bit field descriptions

ACTLR\_EL2 is a 64-bit register, and is part of:

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

**Figure 3-2: ACTLR\_EL2 bit assignments**



#### RES0, [63:13]

RES0 Reserved.

#### CLUSTERPMUEN, [12]

Performance Management Registers enable. The possible values are:

- |   |  |
|---|--|
| 0 | CLUSTERPM* registers are not write-accessible from a lower Exception level. This is the reset value. |
| 1 | CLUSTERPM* registers are write-accessible from EL1 Non-secure if they are write-accessible from EL2. |

**SMEN, [11]**

Scheme Management Registers enable. The possible values are:

- |   |   |
|---|---|
| 0 | Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible from EL1 Non-secure. This is the reset value.          |
| 1 | Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible from EL1 Non-secure if they are write-accessible from EL2. |

**TSIDEN, [10]**

Thread Scheme ID Register enable. The possible values are:

- |   |  |
|---|--|
| 0 | Register CLUSTERTHREADSID is not write-accessible from EL1 Non-secure. This is the reset value.          |
| 1 | Register CLUSTERTHREADSID is write-accessible from EL1 Non-secure if they are write-accessible from EL2. |

**RES0, [9:8]**

<b>RES0</b>	Reserved.
-------------	-----------

**PWREN, [7]**

Power Control Registers enable. The possible values are:

- |   |  |
|---|--|
| 0 | Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write-accessible from EL1 Non-secure. This is the reset value.          |
| 1 | Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible from EL1 Non-secure if they are write-accessible from EL2. |

**RES0, [6]**

<b>RES0</b>	Reserved.
-------------	-----------

**ERXPFGEN, [5]**

Error Record Registers enable. The possible values are:

- |   |   |
|---|---|
| 0 | ERXPFG* are not write-accessible from EL1 Non-secure. This is the reset value.          |
| 1 | ERXPFG* are write-accessible from EL1 Non-secure if they are write-accessible from EL2. |

**RES0, [4:2]**

<b>RES0</b>	Reserved.
-------------	-----------

**ECTLREN, [1]**

Extended Control Registers enable. The possible values are:

- |   |   |
|---|---|
| 0 | CPUECTLR and CLUSTERECTLR are not write-accessible from EL1 Non-secure. This is the reset value.          |
| 1 | CPUECTLR and CLUSTERECTLR are write-accessible from EL1 Non-secure if they are write-accessible from EL2. |

**ACTLREN, [0]**

Auxiliary Control Registers enable. The possible values are:

- |   |   |
|---|---|
| 0 | CPUACTLR and CLUSTERACTLR are not write-accessible from EL1 Non-secure. This is the reset value.          |
| 1 | CPUACTLR and CLUSTERACTLR are write-accessible from EL1 Non-secure if they are write-accessible from EL2. |

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

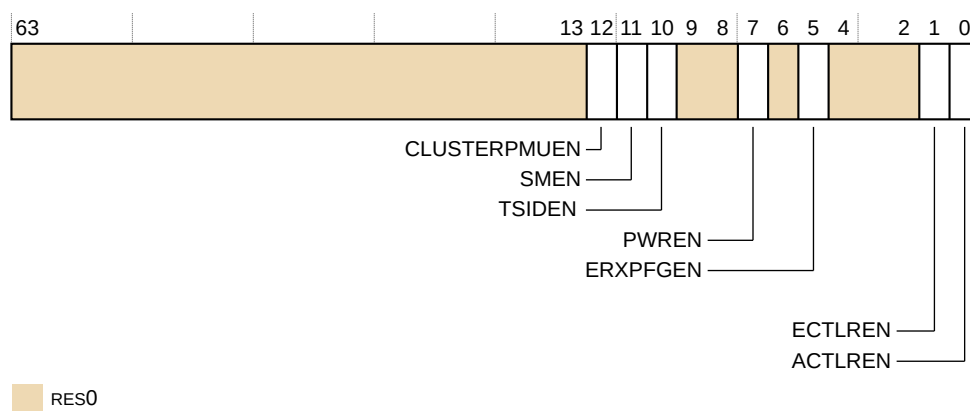
**3.1.7 ACTLR\_EL3, Auxiliary Control Register, EL3**

The ACTLR\_EL3 provides **IMPLEMENTATION DEFINED** configuration and control options for EL3.

**Bit field descriptions**

ACTLR\_EL3 is a 64-bit register, and is part of:

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

**Figure 3-3: ACTLR\_EL3 bit assignments**



**RES0, [63:13]**

**RES0** Reserved.

**CLUSTERPMUEN, [12]**

Performance Management Registers enable. The possible values are:

- |   |  |
|---|--|
| 0 | CLUSTERPM* registers are not write-accessible from a lower Exception level. This is the reset value. |
| 1 | CLUSTERPM* registers are write-accessible from EL2 and EL1 Secure.                                   |

**SMEN, [11]**

Scheme Management Registers enable. The possible values are:

- |   |  |
|---|--|
| 0 | Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible from EL2 and EL1 Secure. This is the reset value. |
| 1 | Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible from EL2 and EL1 Secure.                              |

**TSIDEN, [10]**

Thread Scheme ID Register enable. The possible values are:

- |   |   |
|---|---|
| 0 | Register CLUSTERTHREADSID is not write-accessible from EL2 and EL1 Secure. This is the reset value. |
| 1 | Register CLUSTERTHREADSID is write-accessible from EL2 and EL1 Secure.                              |

**RES0, [9:8]**

**RES0** Reserved.

**PWREN, [7]**

Power Control Registers enable. The possible values are:

- |   |   |
|---|---|
| 0 | Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write-accessible from EL2 and EL1 Secure. This is the reset value. |
| 1 | Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible from EL2 and EL1 Secure.                              |

**RES0, [6]**

**RES0** Reserved.

**ERXPFGEN, [5]**

Error Record Registers enable. The possible values are:

0	ERXPFG* are not write-accessible from EL2 and EL1 Secure. This is the reset value.
1	ERXPFG* are write-accessible from EL2 and EL1 Secure.

**RES0, [4:2]**

<b>RES0</b>	Reserved.
-------------	-----------

**ECTLREN, [1]**

Extended Control Registers enable. The possible values are:

0	CPUECTLR and CLUSTERECTLR are not write-accessible from EL2 and EL1 Secure. This is the reset value.
1	CPUECTLR and CLUSTERECTLR are write-accessible from EL2 and EL1 Secure.

**ACTLREN, [0]**

Auxiliary Control Registers enable. The possible values are:

0	CPUACTLR and CLUSTERACTLR are not write-accessible from EL2 and EL1 Secure. This is the reset value.
1	CPUACTLR and CLUSTERACTLR are write-accessible from EL2 and EL1 Secure.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.8 AFSR0\_EL1, Auxiliary Fault Status Register 0, EL1**

AFSR0\_EL1 provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions that are taken to EL1. In the Cortex®-A65AE core, no additional information is provided for these exceptions. Therefore this register is not used.

**3.1.9 AFSR0\_EL2, Auxiliary Fault Status Register 0, EL2**

AFSR0\_EL2 provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions that are taken to EL2.

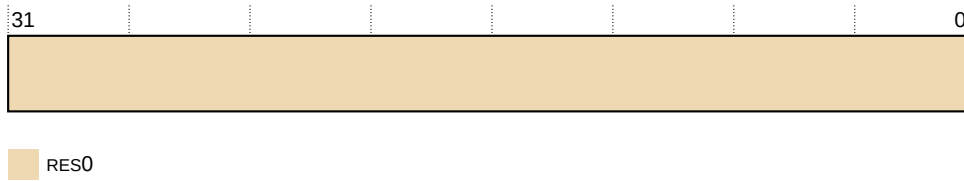
**Bit field descriptions**

AFSR0\_EL2 is a 32-bit register, and is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.

- The **IMPLEMENTATION DEFINED** functional group.

**Figure 3-4: AFSR0\_EL2 bit assignments**



#### RES0, [31:0]

Reserved, **RES0**.

#### Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.10 AFSR0\_EL3, Auxiliary Fault Status Register 0, EL3

AFSR0\_EL3 provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions that are taken to EL3. In the Cortex®-A65AE core, no additional information is provided for these exceptions. Therefore this register is not used.

### 3.1.11 AFSR1\_EL1, Auxiliary Fault Status Register 1, EL1

AFSR1\_EL1 provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions that are taken to EL1. This register is not used in Cortex®-A65AE.

### 3.1.12 AFSR1\_EL2, Auxiliary Fault Status Register 1, EL2

AFSR1\_EL2 provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions that are taken to EL2. This register is not used in the Cortex®-A65AE core.

### 3.1.13 AFSR1\_EL3, Auxiliary Fault Status Register 1, EL3

AFSR1\_EL3 provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions that are taken to EL3. This register is not used in the Cortex®-A65AE core.

### 3.1.14 AIDR\_EL1, Auxiliary ID Register, EL1

AIDR\_EL1 provides **IMPLEMENTATION DEFINED** identification information. This register is not used in the A65AE core.

### 3.1.15 AMAIR\_EL1, Auxiliary Memory Attribute Indirection Register, EL1

AMAIR\_EL1 provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by MAIR\_EL1. This register is not used in the Cortex®-A65AE core.

### 3.1.16 AMAIR\_EL2, Auxiliary Memory Attribute Indirection Register, EL2

AMAIR\_EL2 provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by MAIR\_EL2. This register is not used in the Cortex®-A65AE core.

### 3.1.17 AMAIR\_EL3, Auxiliary Memory Attribute Indirection Register, EL3

AMAIR\_EL3 provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by MAIR\_EL3. This register is not used in the Cortex®-A65AE core.

### 3.1.18 CCSIDR\_EL1, Cache Size ID Register, EL1

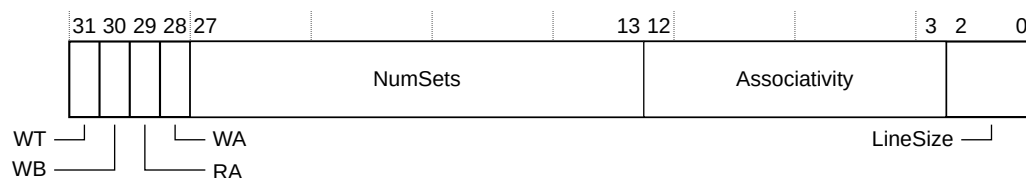
The CCSIDR\_EL1 provides information about the architecture of the currently selected cache.

#### Bit field descriptions

CCSIDR\_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

**Figure 3-5: CCSIDR\_EL1 bit assignments**



#### WT, [31]

Indicates whether the selected cache level supports Write-Through:

0 Cache Write-Through is not supported at any level.

For more information about encoding, see [CCSIDR\\_EL1 encodings](#) on page 110.

**WB, [30]**

Indicates whether the selected cache level supports Write-Back. Permitted values are:

0	Write-Back is not supported.
1	Write-Back is supported.

For more information about encoding, see [CCSIDR\\_EL1 encodings](#) on page 110.

**RA, [29]**

Indicates whether the selected cache level supports read-allocation. Permitted values are:

0	Read-allocation is not supported.
1	Read-allocation is supported.

For more information about encoding, see [CCSIDR\\_EL1 encodings](#) on page 110.

**WA, [28]**

Indicates whether the selected cache level supports write-allocation. Permitted values are:

0	Write-allocation is not supported.
1	Write-allocation is supported.

For more information about encoding, see [CCSIDR\\_EL1 encodings](#) on page 110.

**NumSets, [27:13]**

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

For more information about encoding, see [CCSIDR\\_EL1 encodings](#) on page 110.

**Associativity, [12:3]**

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

For more information about encoding, see [CCSIDR\\_EL1 encodings](#) on page 110.

**LineSize, [2:0]**

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

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

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

For more information about encoding, see [CCSIDR\\_EL1 encodings](#) on page 110.

## Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## CCSIDR\_EL1 encodings

The following table shows the individual bit field and complete register encodings for the CCSIDR\_EL1.

**Table 3-14: CCSIDR encodings**

CSSELR		Cache	Size	Complete register encoding	Register bit field encoding						
Level	InD				WT	WB	RA	WA	NumSets	Associativity	LineSize
0b000	0b0	L1 Data cache	32KB	0x700FE01A	0x0	0x1	0x1	0x1	0x007F	0x003	0x2
			64KB	0x701FE01A	0x0	0x1	0x1	0x1	0x00FF	0x003	0x2
0b000	0b1	L1 Instruction cache	32KB	0x200FE01A	0x0	0x0	0x1	0x0	0x007F	0x003	0x2
			64KB	0x201FE01A	0x0	0x0	0x1	0x0	0x00FF	0x003	0x2
0b001	0b0	L2 cache	64KB	0x701FE01A	0x0	0x1	0x1	0x1	0x00FF	0x003	0x2
			128KB	0x703FE01A	0x0	0x1	0x1	0x1	0x01FF	0x003	0x2
			256KB	0x707FE01A	0x0	0x1	0x1	0x1	0x03FF	0x003	0x2
0b001	0b1	Reserved	-	-	-	-	-	-	-	-	-
0b010	0b0	L3 cache	512KB	0x703FE07A	0x0	0x1	0x1	0x1	0x01FF	0x00F	0x2
			1MB	0x707FE07A	0x0	0x1	0x1	0x1	0x03FF	0x00F	0x2
			2MB	0x70FFE07A	0x0	0x1	0x1	0x1	0x07FF	0x00F	0x2
			4MB	0x71FFE07A	0x0	0x1	0x1	0x1	0x0FFF	0x00F	0x2
0b010	0b1	Reserved	-	-	-	-	-	-	-	-	-
0b0101 - 0b1111		Reserved	-	-	-	-	-	-	-	-	-



If no per-core L2 cache is present the core uses L3 cache as L2 (cluster L2 cache) and the L3 encodings apply.

### 3.1.19 CLIDR\_EL1, Cache Level ID Register, EL1

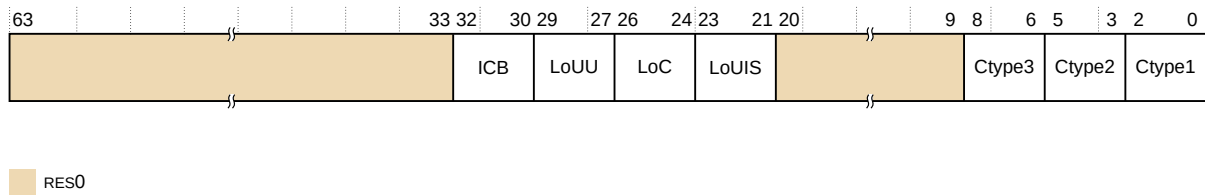
The CLIDR\_EL1 identifies the type of cache, or caches, which are implemented at each level, up to a maximum of seven levels.

It also identifies the *Level of Coherency* (LoC) and *Level of Unification* (LoU) for the cache hierarchy.

#### Bit field descriptions

CLIDR\_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

**Figure 3-6: CLIDR\_EL1 bit assignments****RES0, [63:33]**

**RES0** Reserved.

**ICB, [32:30]**

Inner cache boundary. This field indicates the boundary between the inner and the outer domain:

**0b001**

Neither per-core L2 or cluster cache is present.

**0b010**

Either per-core L2 or cluster L2 cache is present, but not both.

**0b011**

Both per-core L2 and cluster L3 caches are present.

**LoUU, [29:27]**

Indicates the Level of Unification Uniprocessor for the cache hierarchy:

**0b000** No levels of cache need to be cleaned or invalidated when cleaning or invalidating to the Point of Unification. This is the value if no caches are configured.

**LoC, [26:24]**

Indicates the Level of Coherency for the cache hierarchy:

**0b001**

Neither per-core L2 or cluster cache is present.

**0b010**

Either per-core L2 or cluster L2 cache is present, but not both.

**0b011**

Both per-core L2 and cluster L3 caches are present.

**LoUIS, [23:21]**

Indicates the *Level of Unification Inner Shareable* (LoUIS) for the cache hierarchy.

**0b000** No cache level needs cleaning to Point of Unification.

**RES0, [20:9]**

No cache at levels L7 down to L4.

RES0      Reserved.

**Ctype3, [8:6]**

Indicates the type of cache if the core implements L3 cache. If present, unified instruction and data caches at level 3:

0b100      Both per-core L2 and cluster L3 caches are present.  
0b000      All other options.

If Ctype2 has a value of 0b000, then the value of Ctype3 must be **IGNORED**.

**Ctype2, [5:3]**

Indicates the type of unified instruction and data caches at level 2:

0b100      Either per-core L2 or cluster L2 cache is present.  
0b000      All other options.

**Ctype1, [2:0]**

Indicates the type of cache which is implemented at L1:

0b011      Separate instruction and data caches at L1.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

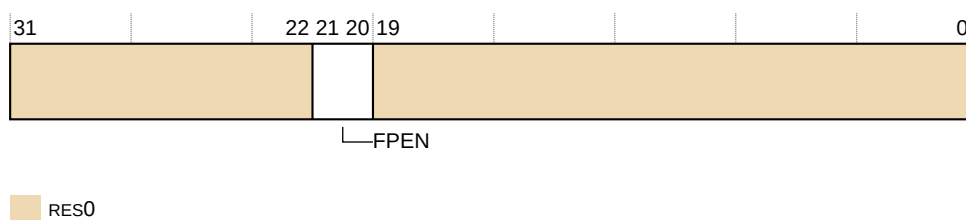
**3.1.20 CPACR\_EL1, Architectural Feature Access Control Register, EL1**

The CPACR\_EL1 controls access to Advanced SIMD and floating-point functionality from ELO, EL1, and EL3.

**Bit field descriptions**

CPACR\_EL1 is a 32-bit register, and is part of the Other system control registers functional group.

**Figure 3-7: CPACR\_EL1 bit assignments**





**RES0, [31:22]**

**RES0** Reserved.

**FPEN, [21:20]**

Traps instructions that access registers associated with Advanced SIMD and floating-point execution to trap to EL1 when executed from EL0 or EL1. The possible values are:

0b00	Trap any instruction in EL0 or EL1 that uses registers associated with Advanced SIMD and floating-point execution. The reset value is 0b00.
0b01	Trap any instruction in EL0 that uses registers associated with Advanced SIMD and floating-point execution. Instructions in EL1 are not trapped.
0b10	Trap any instruction in EL0 or EL1 that uses registers associated with Advanced SIMD and floating-point execution.
0b11	No instructions are trapped.

This field is **RES0** if Advanced SIMD and floating-point are not implemented.

**RAZ/WI, [19:0]**

**RAZ/  
WI** Read as zero, write ignore.

**Configurations**

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.21 CPTR\_EL2, Architectural Feature Trap Register, EL2**

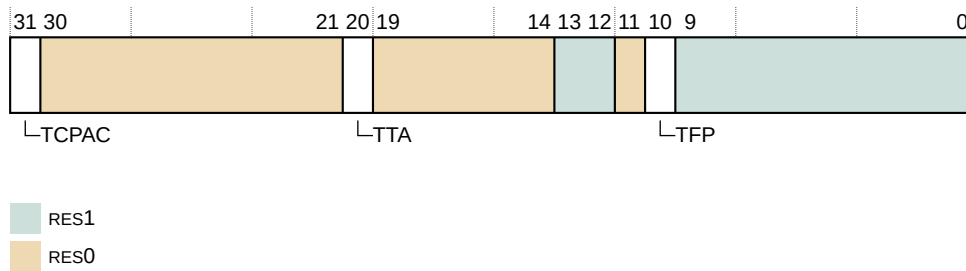
The CPTR\_EL2 controls trapping to EL2 for accesses to CPACR, Trace functionality and registers associated with Advanced SIMD and floating-point execution. It also controls EL2 access to this functionality.

**Bit field descriptions**

CPTR\_EL2 is a 32-bit register, and is part of the Virtualization registers functional group.

When `HCR_EL2.E2H == 0`:

**Figure 3-8: CPTR\_EL2 bit assignments**



### TCPAC, [31]

Traps direct access to CPACR\_EL1 from Non-secure EL1 to EL2. The possible values are:

- |   |  |
|---|--|
| 0 | Access to CPACR is not trapped. This is the reset value. |
| 1 | Non-secure EL1 accesses to CPACR_EL1 are trapped to EL2. |



CPACR\_EL1 is not accessible at EL0.

### RES0, [30:21]

**RES0** Reserved.

### TTA, [20]

Trap Trace Access. Traps Non-secure System register accesses to all implemented trace registers to EL2, from both Execution states.

- |   |   |
|---|---|
| 0 | No accesses are trapped. This is the reset value.   |
| 1 | Any attempt at EL2, or Non-secure EL0 or EL1, to execute a System register access to an implemented trace register is trapped to EL2. |

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

If System register access to the trace functionality is not supported, this bit is RES0.

### RES0, [19:14]

**RES0** Reserved.

RES1, [13:12]

RES1      Reserved.

RES0, [11]

RES0      Reserved.

TFP, [10]

Traps instructions that access registers associated with Advanced SIMD and floating-point execution from a lower Exception level to EL2, unless trapped to EL1. The possible values are:

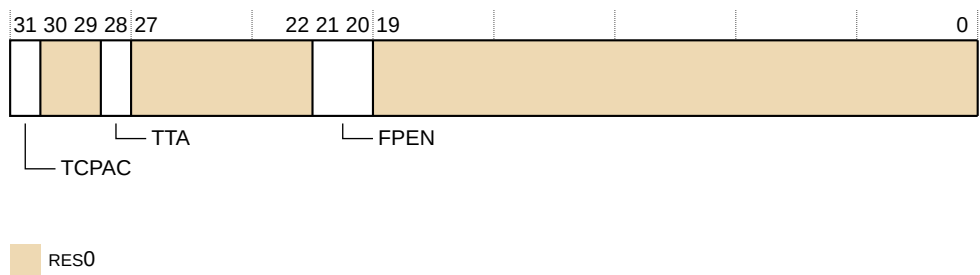
- |   |   |
|---|---|
| 0 | Does not cause any instruction to be trapped. This is the reset value.  |
| 1 | Causes any instructions that use the registers that are associated with Advanced SIMD or floating-point execution to be trapped. This is always the value if the Advanced SIMD and floating-point support is not implemented. |

RES1, [9:0]

RES1      Reserved.

When HCR\_EL2.E2H == 1:

Figure 3-9: CPTR\_EL2 bit assignments



TCPAC, [31]

When HCR\_EL2.TGE is 0, traps Non-secure EL1 accesses to CPACR\_EL1 to EL2, from both Execution states. The possible values are:

- |   |  |
|---|--|
| 0 | Access to CPACR_EL1 is not trapped. This is the reset value. |
| 1 | Non-secure EL1 accesses to CPACR_EL1 are trapped to EL2.     |



CPACR\_EL1 is not accessible at EL0.

**RES0, [30:29]**

**RES0** Reserved.

**TTA, [28]**

Trap Trace Access. Traps Non-secure System register accesses to all implemented trace registers to EL2, from both Execution states.

- |   |   |
|---|---|
| 0 | No accesses are trapped. This is the reset value.   |
| 1 | Any attempt at EL2, or Non-secure EL0 or EL1, to execute a System register access to an implemented trace register is trapped to EL2. |

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

If System register access to the trace functionality is not supported, this bit is RES0.

**RES0, [27:22]**

**RES0** Reserved.

**FPEN, [21:20]**

Traps EL2, Non-secure EL0 and, when HCR\_EL2.TGE is 0, Non-secure EL1 accesses to the Advanced SIMD and floating-point registers to EL2, from both Execution states. The possible values are:

- |      |   |
|------|---|
| 0b00 | This control causes any instructions at Non-secure EL0, EL1, or EL2 that use the registers associated with Advanced SIMD and floating-point execution to be trapped, unless HCR_EL2.TGE is 0 and they are trapped by CPACR_EL1.FPEN, CPACR_EL1.ZEN, or CPTR_EL2.ZEN. This is the reset value.             |
| 0b01 | When HCR_EL2.TGE is 0, this control does not cause any instructions to be trapped.<br>When HCR_EL2.TGE is 1, this control causes instructions at Non-secure EL0 that use the registers associated with Advanced SIMD and floating-point execution to be trapped, unless they are trapped by CPTR_EL2.ZEN. |
| 0b10 | This control causes any instructions at Non-secure EL0, EL1, or EL2 that use the registers associated with Advanced SIMD and floating-point execution to be trapped, unless HCR_EL2.TGE is 0 and they are trapped by CPACR_EL1.FPEN, CPACR_EL1.ZEN, or CPTR_EL2.ZEN.                                      |
| 0b11 | No instructions are trapped.  |

**RES0, [19:18]**

**RES0** Reserved.

**RES0, [17:16]**

**RES0** Reserved.

**RES0, [15:0]**

**RES0** Reserved.

**Configurations**

RW fields in this register reset to **UNKNOWN** values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.22 CPTR\_EL3, Architectural Feature Trap Register, EL3**

The CPTR\_EL3 controls trapping to EL3 of access to CPACR\_EL1, CPTR\_EL2, trace functionality and registers associated with Advanced SIMD and floating-point execution.

It also controls EL3 access to trace functionality and registers associated with Advanced SIMD and floating-point execution.

**Bit field descriptions**

CPTR\_EL3 is a 32-bit register, and is part of the Security registers functional group.

**Figure 3-10: CPTR\_EL3 bit assignments****RAZ/WI, [31:11]**

**RAZ/  
WI** Read-As-Zero, write ignore.

**TFP, [10]**

This causes instructions that access the registers that are associated with Advanced SIMD or floating-point execution to trap to EL3 when executed from any Exception level, unless trapped to EL1 or EL2. The possible values are:

- 0 Does not cause any instruction to be trapped. This is the reset value.
- 1 Any attempt at any Exception level to execute an instruction that uses the registers that are associated with SVE, Advanced SIMD and floating-point is trapped to EL3, subject to the exception prioritization rules.

**RAZ/WI, [9:0]**

**RAZ/**  
**WI**      Read-As-Zero, write ignore.

**Configurations**

There are no configuration notes.

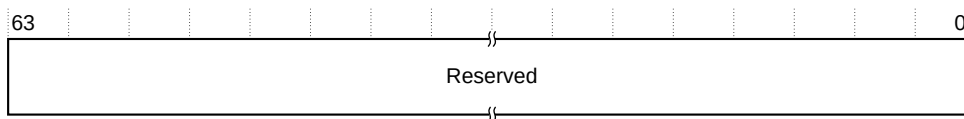
Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.23 CPUACTLR\_EL1, CPU Auxiliary Control Register, EL1**

The CPUACTLR\_EL1 provides **IMPLEMENTATION DEFINED** configuration and control options for the core.

**Bit field descriptions**

CPUACTLR\_EL1 is a 64-bit register, and is part of the **IMPLEMENTATION DEFINED** registers functional group.

**Figure 3-11: CPUACTLR\_EL1 bit assignments****Reserved, [63:0]**

Reserved for Arm® internal use.

**Configurations**

CPUACTLR\_EL1 is common to the Secure and Non-secure states.

**Usage constraints****Accessing the CPUACTLR\_EL1**

The CPU Auxiliary Control Register can be written only when the system is idle. Arm recommends that you write to this register after a Cold reset, before the MMU is enabled. While each thread appears to have its own CPUACTLR\_EL1, the register is physically shared. Writes from one thread are visible to the other thread. Arm recommends that the register only be updated when software synchronization between the threads guarantees exclusive access to the register.

Setting many of these bits can cause significantly lower performance on your code. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_0	11	000	1111	0001	000

### Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C1_0	x	x	0	-	RW	n/a	RW
S3_0_C15_C1_0	x	0	1	-	RW	RW	RW
S3_0_C15_C1_0	x	1	1	-	n/a	RW	RW

This register is write-accessible in EL1 on either of these conditions:

- ACTLR\_EL3.CPUACTLR\_EN == 1 && ACTLR\_EL2.CPUACTLR\_EN == 1.
- ACTLR\_EL3.CPUACTLR\_EN == 1 && SCR.NS == 0.

This register is write-accessible in EL2 if ACTLR\_EL3.CPUACTLR\_EN == 1.

If Write-Access is not possible, then trap to the lowest Exception level that denied the access (EL2 or EL3).

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

### Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.24 CPUCFR\_EL1, CPU Configuration Register, EL1

The CPUCFR\_EL1 provides configuration information for the core.

#### Bit field descriptions

CPUCFR\_EL1 is a 32-bit register, and is part of the **IMPLEMENTATION DEFINED** registers functional group.

This register is read-only, Write Ignore.

**Figure 3-12: CPUCFR\_EL1 bit assignments**



**RES0**

#### RES0, [31:2]

**RES0** Reserved.

#### ECC, [1:0]

Indicates whether ECC is present or not. The value is:

**01**

ECC is present.

#### Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

#### Usage constraints

##### Accessing the CPUCFR\_EL1

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>, <systemreg>
```

To access the CPUCFR\_EL1:

```
MRS <Xt>, CPUCFR_EL1 ; Read CPUCFR_EL1 into Xt
```

This syntax is encoded with the following settings in the instruction encoding:



<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_CO_0	11	000	1111	0000	000

### Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_CO_0	x	x	0	-	RO	n/a	RO
S3_0_C15_CO_0	x	0	1	-	RO	RO	RO
S3_0_C15_CO_0	x	1	1	-	n/a	RO	RO

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

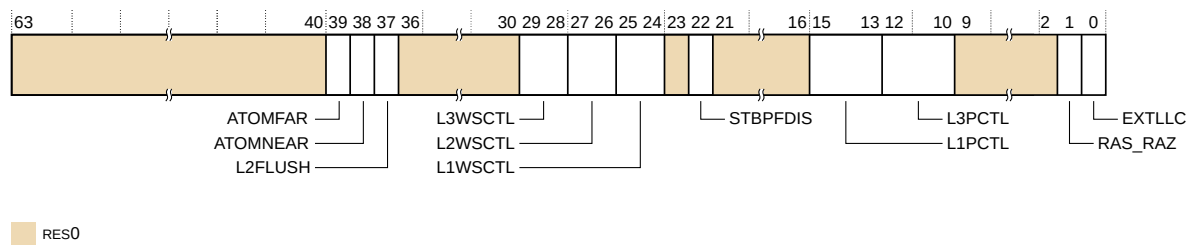
## 3.1.25 CPUECTLR\_EL1, CPU Extended Control Register, EL1

The CPUECTLR\_EL1 provides extra **IMPLEMENTATION DEFINED** configuration and control options for the core.

### Bit field descriptions

CPUECTLR\_EL1 is a 64-bit register, and is part of the 64-bit registers functional group.

**Figure 3-13: CPUECTLR\_EL1 bit assignments**



### RES0, [63:40]

RES0 Reserved.

### ATOMIC, [39:38]

Controls behavior of both cacheable load atomic instructions (including SWP and CAS) and store atomic instructions that do not hit in the L1 cache in the unique state. The possible values are:

00 Cacheable load atomic instructions (including SWP and CAS) will make up one fill request to perform as near atomics. They will otherwise perform as far atomics. Cacheable store atomic instructions are

- performed as near atomics if they hit the cache in a unique state. Otherwise, they are performed as far atomics. This is the reset value.
- 01 All cacheable atomic instructions are forced to be executed as near atomics in the L1 cache.
  - 10 All cacheable atomic instructions are forced to be executed as far atomics in the L3 cache or beyond.
  - 11 Cacheable load atomic instructions (including SWP and CAS) are forced to be executed as near atomics. Store atomics are performed as near atomics if they hit in the cache in a unique state; otherwise, they are performed as far atomics.

**L2FLUSH, [37]**

Controls handling of clean lines during core powerdown sequences. The possible values are:

- 0 During a core powerdown sequence, clean lines in the L1 and L2 caches are allocated into the L3 cache. This can improve performance if it is known that the data is likely to be used soon by another core. This is the reset value.
- 1 During a core powerdown sequence, clean lines in the L1 and L2 caches are discarded and do not allocate into the L3 cache.

**RES0, [36:30]**

**RES0** Reserved.

**L3WSCTL, [29:28]**

Write streaming no-L3-allocate threshold. The possible values are:

- 00 128th consecutive streaming cache line does not allocate in the L1, L2, or L3 cache.
- 01 1024th consecutive streaming cache line does not allocate in the L1, L2, or L3 cache. This is the reset value.
- 10 4096th consecutive streaming cache line does not allocate in the L1, L2, or L3 cache.
- 11 Disables streaming. All Write-Allocate lines allocate in the L1, L2, or L3 cache.

**L2WSCTL, [27:26]**

Write streaming no-L2-allocate threshold. The possible values are:

- 00 16th consecutive streaming cache line does not allocate in the L1 or L2 cache.
- 01 128th consecutive streaming cache line does not allocate in the L1 or L2 cache. This is the reset value.
- 10 512th consecutive streaming cache line does not allocate in the L1 or L2 cache.
- 11 Disables streaming. All Write-Allocate lines allocate in the L1 or L2 cache.

**L1WSCTL, [25:24]**

Write streaming no-L1-allocate threshold. The possible values are:

00	4th consecutive streaming cache line does not allocate in the L1 cache. This is the reset value.
01	64th consecutive streaming cache line does not allocate in the L1 cache.
10	128th consecutive streaming cache line does not allocate in the L1 cache.
11	Disables streaming. All Write-Allocate lines allocate in the L1 cache.

**RES0, [23]**

**RES0** Reserved.

**STBPFDIS, [22]**

Disable *Store Buffer* STB accesses from being tracked and trained by the hardware prefetcher. The possible values are:

0	STB accesses are tracked and trained by the hardware prefetcher. Hardware prefetch requests may be generated as a result of STB generated line fills. This is the reset value.
1	STB accesses are not tracked and trained by the prefetcher. No hardware prefetch requests will be generated as a result of STB generated line fills.

**RES0, [21:16]**

**RES0** Reserved.

**L1PCTL, [15:13]**

L1 Data prefetch control. The value of the L1PCTL field determines the maximum number of outstanding data prefetches allowed in the L1 memory system (not counting the data prefetches generated by software load/*PLD* instructions).

000	Prefetch disabled.
001	1 outstanding prefetch allowed.
010	2 outstanding prefetches allowed.
011	3 outstanding prefetches allowed.
100	4 outstanding prefetches allowed.
101	5 outstanding prefetches allowed. This is the reset value.
110	6 outstanding prefetches allowed.
111	7 outstanding prefetches allowed.

**L3PCTL, [12:10]**

L3 Data prefetch control. The value of the L3PCTL field determines the approximate distance between the L1 prefetcher and requests sent to the L3 memory system. Increasing this distance may improve performance on systems with higher latency to main memory, but increasing it too far can reduce performance.

000	Fetch 16 lines ahead.
001	Fetch 32 lines ahead.
010	Reserved.
011	Reserved.
100	Disable L3 prefetching.
101	Fetch 2 lines ahead.
110	Fetch 4 lines ahead.
111	Fetch 8 lines ahead. This is the reset value.

**RES0, [9:2]**

<b>RES0</b>	Reserved.
-------------	-----------

**RAS\_RAZ, [1]**

Force RAS register reads to Read-As-Zero. The possible values are:

0	A read of a RAS register returns the current value of the register.
1	A read of a RAS register returns a 0.

Set this bit to 1 to force reads of RAS registers to return 0 in cases where a fault such as **cpu\_errmisc0** might alter the register contents. This behavior prevents a read from causing divergence, after a fault where the only divergence is the updating one or more of the RAS register bits.

**EXTLLC, [0]**

0	Indicates that an external Last-level cache is present in the system, and that the DataSource field on the master CHI interface indicates when data is returned from the LLC. This is used to control how the LL_CACHE* PMU events count.
---	---

**Configurations**

This register has no configuration notes.

**Usage constraints****Accessing the CPUECTLR\_EL1**

The CPUECTLR\_EL1 can be written dynamically.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written with the MRS instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
CPUECTLR_EL1	11	000	1111	0001	100

### Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CPUECTLR_EL1	x	x	0	-	RW	n/a	RW
CPUECTLR_EL1	x	0	1	-	RW	RW	RW
CPUECTLR_EL1	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

### Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

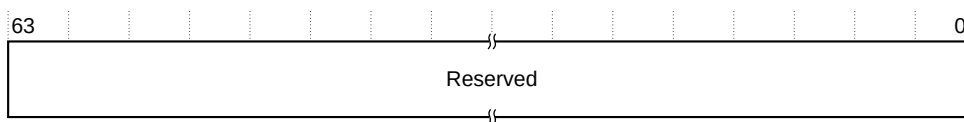
## 3.1.26 CPUPCR\_EL3, CPU Private Control Register, EL3

The CPUPCR\_EL3 provides **IMPLEMENTATION DEFINED** configuration and control options for the core.

### Bit field descriptions

CPUPCR\_EL3 is a 64-bit register, and is part of the **IMPLEMENTATION DEFINED** registers functional group.

**Figure 3-14: CPUPCR\_EL3 bit assignments**



### Reserved, [63:0]

Reserved for Arm® internal use.

### Configurations

CPUPCR\_EL3 is only accessible in Secure state.

## Usage constraints

### Accessing the CPUPCR\_EL3

The CPUPCR\_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause **UNPREDICTABLE** behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_8_1	11	110	1111	1000	001

### Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_1	x	x	0	-	-	n/a	RW
S3_6_C15_8_1	x	0	1	-	-	-	RW
S3_6_C15_8_1	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

### Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

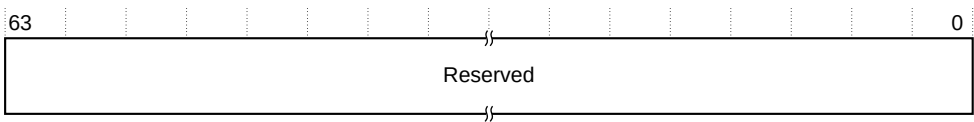
### 3.1.27 CPUPMR\_EL3, CPU Private Mask Register, EL3

The CPUPMR\_EL3 provides **IMPLEMENTATION DEFINED** configuration and control options for the core.

#### Bit field descriptions

CPUPMR\_EL3 is a 64-bit register, and is part of the **IMPLEMENTATION DEFINED** registers functional group.

Figure 3-15: CPUPMR\_EL3 bit assignments



#### Reserved, [63:0]

Reserved for Arm® internal use.

#### Configurations

CPUPMR\_EL3 is only accessible in Secure state.

#### Usage constraints

##### Accessing the CPUPMR\_EL3

The CPUPMR\_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause unpredictable behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_8_3	11	110	1111	1000	011

## Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_3	x	x	0	-	-	n/a	RW
S3_6_C15_8_3	x	0	1	-	-	-	RW
S3_6_C15_8_3	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

## Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

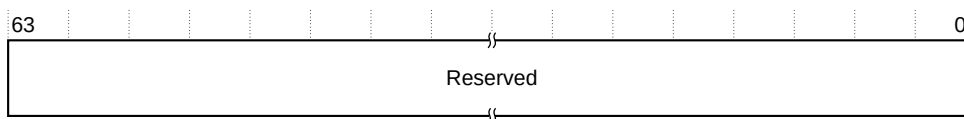
## 3.1.28 CPUPOR\_EL3, CPU Private Operation Register, EL3

The CPUPOR\_EL3 provides **IMPLEMENTATION DEFINED** configuration and control options for the core.

### Bit field descriptions

CPUPOR\_EL3 is a 64-bit register, and is part of the **IMPLEMENTATION DEFINED** registers functional group.

**Figure 3-16: CPUPOR\_EL3 bit assignments**



### Reserved, [63:0]

Reserved for Arm® internal use.

### Configurations

CPUPOR\_EL3 is only accessible in Secure state.

### Usage constraints

#### Accessing the CPUPOR\_EL3

The CPUPOR\_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.



Writing to this register might cause unpredictable behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_8_2	11	110	1111	1000	010

### Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_2	x	x	0	-	-	n/a	RW
S3_6_C15_8_2	x	0	1	-	-	-	RW
S3_6_C15_8_2	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

### Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the Arm® Architecture Reference Manual Armv8, for A-profile architecture.

## 3.1.29 CPUPSELR\_EL3, CPU Private Selection Register, EL3

The CPUPSELR\_EL3 provides **IMPLEMENTATION DEFINED** configuration and control options for the core.

### Bit field descriptions

CPUPSELR\_EL3 is a 32-bit register, and is part of the **IMPLEMENTATION DEFINED** registers functional group.

**Figure 3-17: CPUPSELR\_EL3 bit assignments****Reserved, [31:0]**

Reserved for Arm® internal use.

**Configurations**

CPUPSELR\_EL3 is only accessible in Secure state.

**Usage constraints****Accessing the CPUPSELR\_EL3**

The CPUPSELR\_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause unpredictable behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_8_0	11	110	1111	1000	000

**Accessibility**

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_0	x	x	0	-	-	n/a	RW
S3_6_C15_8_0	x	0	1	-	-	-	RW

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_0	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

### Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 3.1.30 CPUPWRCTLR\_EL1, Power Control Register, EL1

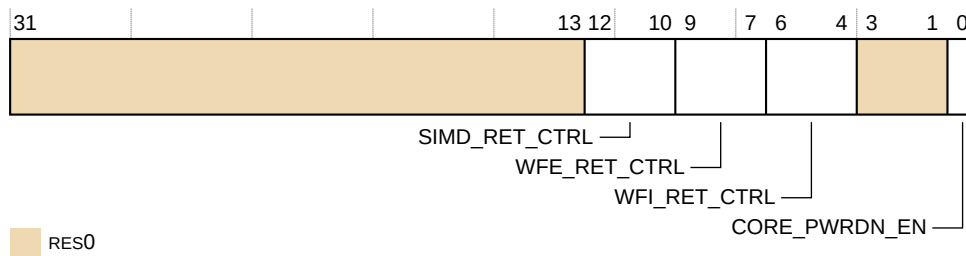
The CPUPWRCTLR\_EL1 provides information about power control support for the core.

### Bit field descriptions

CPUPWRCTLR\_EL1 is a 32-bit register, and is part of the **IMPLEMENTATION DEFINED** registers functional group.

This register resets to value 0x00000000.

**Figure 3-18: CPUPWRCTLR\_EL1 bit assignments**



### RES0, [31:13]

**RES0** Reserved.

### SIMD\_RET\_CTRL, [12:10]

Advanced SIMD and floating-point retention control:

000 Disable the retention circuit. This is the default value, see [CPUPWRCTLR Retention Control Field](#) on page 132 for more retention control options.

### WFE\_RET\_CTRL, [9:7]

CPU WFE retention control:

000 Disable the retention circuit. This is the default value, see [CPUPWRCTLR Retention Control Field](#) on page 132 for more retention control options.

### WFI\_RET\_CTRL, [6:4]

CPU WFI retention control:

000 Disable the retention circuit. This is the default value, see [CPUPWRCTLR Retention Control Field](#) on page 132 for more retention control options.

### RES0, [3:1]

RES0 Reserved.

### CORE\_PWRDN\_EN, [0]

Indicates to the power controller using PACTIVE if the core wants to power down when it enters WFI state.

0 No powerdown requested. This is the reset value.  
1 A power down is requested.

**Table 3-29: CPUPWRCTLR Retention Control Field**

Encoding	Number of counter ticks <sup>2</sup>	Minimum retention entry delay (System counter at 50MHz-10MHz)
000	Disable the retention circuit	Default Condition.
001	2	40ns-200ns
010	8	160ns-800ns
011	32	640ns – 3,200ns
100	64	1,280ns-6,400ns
101	128	2,560ns-12,800ns
110	256	5,120ns-25,600ns
111	512	10,240ns-51,200ns

### Configurations

There are no configuration notes.

### Usage constraints

#### Accessing the CPUPWRCTLR\_EL1

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

```
MRS <Xt>, <systemreg>
```

<sup>2</sup> The number of system counter ticks required before the core signals retention readiness on PACTIVE to the power controller. The core does not accept a retention entry request until this time.

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

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_O_C15_C2_7	11	000	1111	0010	111

### Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_O_C15_C2_7	x	x	0	-	RW	n/a	RW
S3_O_C15_C2_7	x	0	1	-	RW	RW	RW
S3_O_C15_C2_7	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

### Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for exceptions taken to AArch64 state.

Write-Access to this register from EL1 or EL2 depends on the value of bit[7] of ACTLR\_EL2 and ACTLR\_EL3.

## 3.1.31 CSSELR\_EL1, Cache Size Selection Register, EL1

CSSELR\_EL1 selects the current Cache Size ID Register (CCSIDR\_EL1), by specifying:

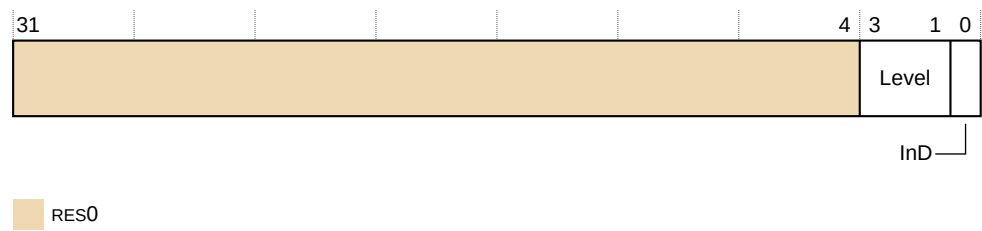
- The required cache level.
- The cache type, either instruction or data cache.

For details of the CCSIDR\_EL1, see [3.1.18 CCSIDR\\_EL1, Cache Size ID Register, EL1](#) on page 108.

### Bit field descriptions

CSSELR\_EL1 is a 32-bit register, and is part of the Identification registers functional group.

Figure 3-19: CSSELR\_EL1 bit assignments



RES0, [31:4]

RES0      Reserved.

Level, [3:1]

Cache level of required cache:

000	L1.
001	L2.
010	L3, if present.

The combination of Level=001 and InD=1 is reserved.

The combinations of Level and InD for 0100 to 1111 are reserved.

InD, [0]

Instruction not Data bit:

0	Data or unified cache.
1	Instruction cache.

The combination of Level=001 and InD=1 is reserved.

The combinations of Level and InD for 0100 to 1111 are reserved.

Configurations

If a cache level is missing but CSSELR\_EL1 selects this level, then a CCSIDR\_EL1 read returns an **UNKNOWN** value.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.32 CTR\_ELO, Cache Type Register, ELO

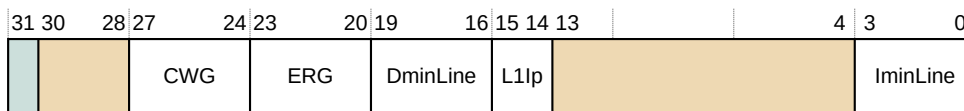
The CTR\_ELO provides information about the architecture of the caches.

#### Bit field descriptions

CTR\_ELO is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

**Figure 3-20: CTR\_ELO bit assignments**



#### RES1, [31]

RES1      Reserved.

#### RES0, [30:28]

RES0      Reserved.

#### CWG, [27:24]

Cache write-back granule.  $\log_2$  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      Cache write-back granule size is 16 words.

#### ERG, [23:20]

Exclusives Reservation Granule.  $\log_2$  of the number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions:

0100      Exclusive reservation granule size is 16 words.

#### DminLine, [19:16]

$\log_2$  of the number of words in the smallest cache line of all the data and unified caches that the core controls:

0100      Smallest data cache line size is 16 words.

**L1Ip, [15:14]**  
Instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache:

10      *Virtually Indexed Physically Tagged* (VIPT).

**RES0, [13:4]**

RES0      Reserved.

**IminLine, [3:0]**  
 $\log_2$  of the number of words in the smallest cache line of all the instruction caches that the core controls.

0100      Smallest instruction cache line size is 16 words.

**Configurations**  
There are no configuration notes.

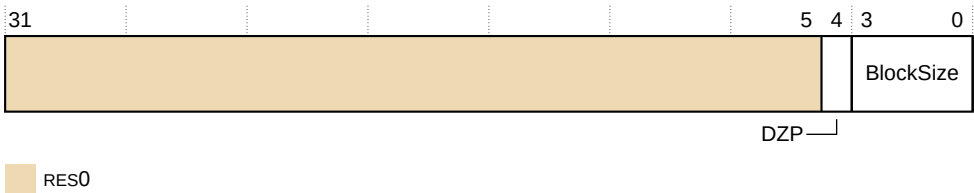
Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

3.1.33 DCZID\_EL0, Data Cache Zero ID Register, EL0

The DCZID\_EL0 indicates the block size that is written with byte values of zero by the `dc zva` (Data Cache Zero by Address) System instruction.

**Bit field descriptions**  
DCZID\_EL0 is a 32-bit register, and is part of the Identification registers functional group.  
This register is read-only.

Figure 3-21: DCZID\_EL0 bit assignments



**RES0, [31:5]**

RES0      Reserved.



**DZP, [4]**

Prohibit the `DC ZVA` instruction:

0	<code>DC ZVA</code> instruction is permitted.
1	<code>DC ZVA</code> instruction is prohibited.

**BlockSize, [3:0]**

$\log_2$  of the block size in words:

0100	The block size is 16 words.
------	-----------------------------

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

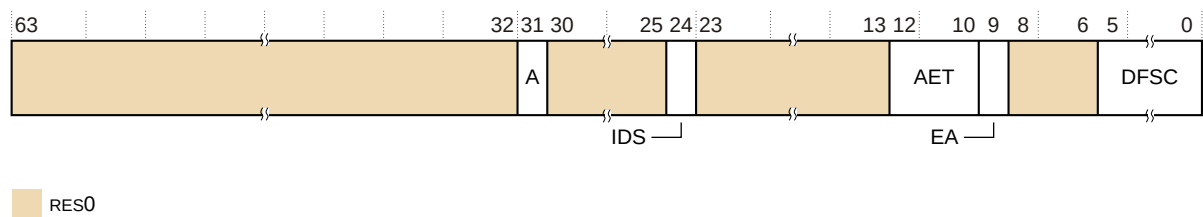
**3.1.34 DISR\_EL1, Deferred Interrupt Status Register, EL1**

The `DISR_EL1` records the SError interrupts consumed by an `ESB` instruction.

**Bit field descriptions**

`DISR_EL1` is a 64-bit register, and is part of the registers *Reliability, Availability, Serviceability* (RAS) functional group.

**Figure 3-22: `DISR_EL1` bit assignments, `DISR_EL1.IDS` is 0**

**RES0, [63:32]**

RES0	Reserved.
------	-----------

**A, [31]**

Set to 1 when `ESB` defers an asynchronous SError interrupt. If the implementation does not include any synchronizable sources of SError interrupt, this bit is **RES0**.

**RES0, [30:25]**

RES0	Reserved.
------	-----------

**IDS, [24]**

Indicates the type of format the deferred SError interrupt uses. The value of this bit is:

0	Deferred error uses architecturally-defined format.
---	---

**RES0, [23:13]**

RES0	Reserved.
------	-----------

**AET, [12:10]**

Asynchronous Error Type. Describes the state of the core after taking an asynchronous Data Abort exception. The value or values are:

0b000	Uncontainable error (UC).
0b001	Unrecoverable error (UEU).
0b010	Restartable error (UEO).
0b011	Recoverable error (UER).
0b110	Corrected error (CE).

All other values are reserved. Reserved values might be defined in a future version of the architecture.



- This field is only valid if IDS == 0b0 and DFSC == 0b010001.
  - The recovery software must also examine any implemented fault records to determine the location and extent of the error.
- 

**EA, [9]**

RES0	Reserved.
------	-----------

**RES0, [8:6]**

RES0	Reserved.
------	-----------

**DFSC, [5:0]**

Data Fault Status Code. The possible values of this field are:

010001	Asynchronous SError interrupt.
--------	--------------------------------

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.35 ERRIDR\_EL1, Error ID Register, EL1

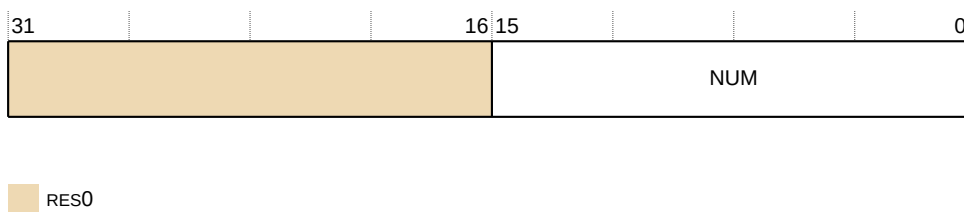
The ERRIDR\_EL1 defines the number of error record registers.

#### Bit field descriptions

ERRIDR\_EL1 is a 32-bit register, and is part of the registers *Reliability, Availability, Serviceability* (RAS) functional group.

This register is read-only.

**Figure 3-23: ERRIDR\_EL1 bit assignments**



#### RES0, [31:16]

**RES0** Reserved.

#### NUM, [15:0]

Number of records that can be accessed through the Error Record System registers.

0x0002 Two records present.

#### Configurations

There are no configuration notes.

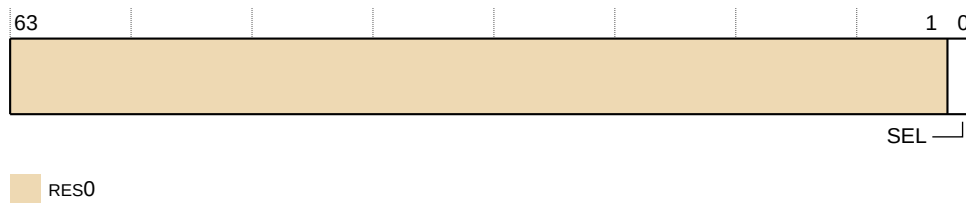
Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.36 ERRSELR\_EL1, Error Record Select Register, EL1

The ERRSELR\_EL1 selects which error record should be accessed through the Error Record System registers. This register is not reset on a Warm reset.

#### Bit field descriptions

ERRSELR\_EL1 is a 64-bit register, and is part of the *Reliability, Availability, Serviceability* (RAS) registers functional group.

**Figure 3-24: ERRSELR\_EL1 bit assignments****RES0, [63:1]**

Reserved, *RES0*.

**SEL, [0]**

Selects which error record should be accessed.

- |   |   |
|---|---|
| 0 | Select error record 0 containing errors from level 1 and level 2 RAMs that are located on the Cortex®-A65AE core.         |
| 1 | Select error record 1 containing errors from level 3 RAMs that are located on the <i>DynamlQ Shared Unit AE</i> (DSU-AE). |

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.37 ERXCTLR\_EL1, Selected Error Record Control Register, EL1**

Register ERXCTLR\_EL1 accesses the ERR<n>CTLR control register for the error record that is selected by ERRSELR\_EL1.SEL.

If ERRSELR\_EL1.SEL==0, then ERXCTLR\_EL1 accesses the ERROCTLR register of the core error record. See [3.2.2 ERROCTLR, Error Record Control Register](#) on page 191.

If ERRSELR\_EL1.SEL==1, then ERXCLTR\_EL1 accesses the ERR1CTLR register of the DSU-AE error record. See the *Arm® DynamlQ™ Shared Unit AE Technical Reference Manual*.

**3.1.38 ERXFR\_EL1, Selected Error Record Feature Register, EL1**

Register ERXFR\_EL1 accesses the ERR<n>FR feature register for the error record that is selected by ERRSELR\_EL1.SEL.

If ERRSELR\_EL1.SEL==0, then ERXFR\_EL1 accesses the ERROFR register of the core error record. See [3.2.3 ERROFR, Error Record Feature Register](#) on page 193.

If `ERRSELR_EL1.SEL==1`, then `ERXFR_EL1` accesses the `ERR1FR` register of the *DynamlQ Shared Unit AE (DSU-AE)* error record. See the *Arm® DynamlQ™ Shared Unit AE Technical Reference Manual*.

### 3.1.39 ERXMISCO\_EL1, Selected Error Record Miscellaneous Register 0, EL1

Register `ERXMISCO_EL1` accesses the `ERR<n>MISCO` register for the error record that is selected by `ERRSELR_EL1.SEL`.

If `ERRSELR_EL1.SEL==0`, then `ERXMISCO_EL1` accesses the `ERR0MISCO` register of the core error record. See [3.2.4 ERR0MISCO, Error Record Miscellaneous Register 0](#) on page 195.

If `ERRSELR_EL1.SEL==1`, then `ERXMISCO_EL1` accesses the `ERR1MISCO` register of the *DynamlQ Shared Unit AE (DSU-AE)* error record. See the *Arm® DynamlQ™ Shared Unit AE Technical Reference Manual*.

### 3.1.40 ERXPFGCDN\_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1

Register `ERXPFGCDN_EL1` accesses the `ERR<n>PFGCND` register for the error record that is selected by `ERRSELR_EL1.SEL`.

If `ERRSELR_EL1.SEL==0`, then `ERXPFGCDN_EL1` accesses the `ERR0PFGCDN` register of the core error record. See [3.2.5 ERR0PFGCDN, Error Pseudo Fault Generation Count Down Register](#) on page 198.

If `ERRSELR_EL1.SEL==1`, then `ERXPFGCDN_EL1` accesses the `ERR1PFGCDNR` register of the *DynamlQ Shared Unit AE (DSU-AE)* error record. See the *Arm® DynamlQ™ Shared Unit AE Technical Reference Manual*.

#### Configurations

There are no configuration notes.

#### Accessing the ERXPFGCDN\_EL1

This register can be read using `MRS` with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using `MSR` with the following syntax:

```
MSR <Xt>, <systemreg>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_O_C15_C2_2	11	000	1111	0010	010

### Accessibility

This register is accessible in software as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_O_C15_C2_2	x	x	0	-	RW	n/a	RW
S3_O_C15_C2_2	x	0	1	-	RW	RW	RW
S3_O_C15_C2_2	x	1	1	-	n/a	RW	RW
n/a	Not accessible. Executing the PE at this Exception level is not permitted.						

### Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for exceptions taken to AArch64 state. Subject to these prioritization rules, the following traps and enables are applicable when accessing this register.

ERXPFGCDN\_EL1 is accessible at EL3 and can be accessible at EL1 and EL2 depending on the value of bit[5] in ACTLR\_EL2 and ACTLR\_EL3. See [3.1.6 ACTLR\\_EL2, Auxiliary Control Register, EL2](#) on page 102 and [3.1.7 ACTLR\\_EL3, Auxiliary Control Register, EL3](#) on page 104.

ERXPFGCDN\_EL1 is *UNDEFINED* at EL0.

If ERXPFGCDN\_EL1 is accessible at EL1 and HCR\_EL2.TERR == 1, then direct reads and writes of ERXPFGCDN\_EL1 at Non-secure EL1 generate a Trap exception to EL2.

If ERXPFGCDN\_EL1 is accessible at EL1 or EL2 and SCR\_EL3.TERR == 1, then direct reads and writes of ERXPFGCDN\_EL1 at EL1 or EL2 generate a Trap exception to EL3.

## 3.1.41 ERXPFGCTL\_EL1, Selected Error Pseudo Fault Generation Control Register, EL1

Register ERXPFGCTL\_EL1 accesses the ERR<n>PFGCTL register for the error record that is selected by ERRSELR\_EL1.SEL.

If ERRSELR\_EL1.SEL==0, then ERXPFGCTL\_EL1 accesses the ERROPFGCTL register of the core error record. See [3.2.6 ERROPFGCTL, Error Pseudo Fault Generation Control Register](#) on page 198.

If ERRSELR\_EL1.SEL==1, then ERXPFGCTL\_EL1 accesses the ERR1PFGCTL register of the *DynamiQ Shared Unit AE (DSU-AE)* error record. See the *Arm® DynamiQ™ Shared Unit AE Technical Reference Manual*.

## Configurations

There are no configuration notes.

## Accessing the ERXPFGCTL\_EL1

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

```
MRS <Xt>, <systemreg>
```

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

```
MSR <Xt>, <systemreg>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_1	11	000	1111	0010	001

## Accessibility

This register is accessible in software as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C2_1	x	x	0	-	RW	n/a	RW
S3_0_C15_C2_1	x	0	1	-	RW	RW	RW
S3_0_C15_C2_1	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

## Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for exceptions taken to AArch64 state. Subject to these prioritization rules, the following traps and enables are applicable when accessing this register.

ERXPFGCTL\_EL1 is accessible at EL3 and can be accessible at EL1 and EL2 depending on the value of bit[5] in ACTLR\_EL2 and ACTLR\_EL3. See [3.1.6 ACTLR\\_EL2, Auxiliary Control Register, EL2](#) on page 102 and [3.1.7 ACTLR\\_EL3, Auxiliary Control Register, EL3](#) on page 104.

ERXPFGCTL\_EL1 is *UNDEFINED* at EL0.

If ERXPFGCTL\_EL1 is accessible at EL1 and HCR\_EL2.TERR == 1, then direct reads and writes of ERXPFGCTL\_EL1 at Non-secure EL1 generate a Trap exception to EL2.

If ERXPFGCTL\_EL1 is accessible at EL1 or EL2 and SCR\_EL3.TERR == 1, then direct reads and writes of ERXPFGCTL\_EL1 at EL1 or EL2 generate a Trap exception to EL3.

### 3.1.42 ERXPFGF\_EL1, Selected Pseudo Fault Generation Feature Register, EL1

Register ERXPFGF\_EL1 accesses the ERR<n>PFGF register for the error record that is selected by ERRSELR\_EL1.SEL.

If ERRSELR\_EL1.SEL==0, then ERXPFGF\_EL1 accesses the ERR0PFGF register of the core error record. See [3.2.7 ERR0PFGF, Error Pseudo Fault Generation Feature Register](#) on page 200.

If ERRSELR\_EL1.SEL==1, then ERXPFGF\_EL1 accesses the ERR1PFGFR register of the *DynamiQ Shared Unit AE* (DSU-AE) error record. See the *Arm® DynamiQ™ Shared Unit AE Technical Reference Manual*.

#### Configurations

This core has no configuration notes.

#### Accessing the ERXPFGF\_EL1

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

```
MRS <Xt>, <systemreg>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_0	11	000	1111	0010	000

#### Accessibility

This register is accessible in software as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C2_0	x	x	0	-	RO	n/a	RO
S3_0_C15_C2_0	x	0	1	-	RO	RO	RO
S3_0_C15_C2_0	x	1	1	-	n/a	RO	RO

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

#### Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for exceptions taken to AArch64 state. Subject to these prioritization rules, the following traps and enables are applicable when accessing this register.

ERXPFGF\_EL1 is accessible at EL3 and can be accessible at EL1 and EL2 depending on the value of bit[5] in ACTLR\_EL2 and ACTLR\_EL3. See [3.1.6 ACTLR\\_EL2, Auxiliary Control](#)



Register, EL2 on page 102 and 3.1.7 ACTLR\_EL3, Auxiliary Control Register, EL3 on page 104.

ERXPFGR\_EL1 is *UNDEFINED* at EL0.

If ERXPFGR\_EL1 is accessible at EL1 and HCR\_EL2.TERR == 1, then direct reads and writes of ERXPFGR\_EL1 at Non-secure EL1 generate a Trap exception to EL2.

If ERXPFGR\_EL1 is accessible at EL1 or EL2 and SCR\_EL3.TERR == 1, then direct reads and writes of ERXPFGR\_EL1 at EL1 or EL2 generate a Trap exception to EL3.

### 3.1.43 ERXSTATUS\_EL1, Selected Error Record Primary Status Register, EL1

Register ERXSTATUS\_EL1 accesses the ERR<n>STATUS primary status register for the error record that is selected by ERRSELR\_EL1.SEL.

If ERRSELR\_EL1.SEL==0, then ERXSTATUS\_EL1 accesses the ERROSTATUS register of the core error record. See 3.2.8 ERROSTATUS, Error Record Primary Status Register on page 202.

If ERRSELR\_EL1.SEL==1, then ERXSTATUS\_EL1 accesses the ERR1STATUS register of the *DynamiQ Shared Unit AE (DSU-AE)* error record. See the *Arm® DynamiQ™ Shared Unit AE Technical Reference Manual*.

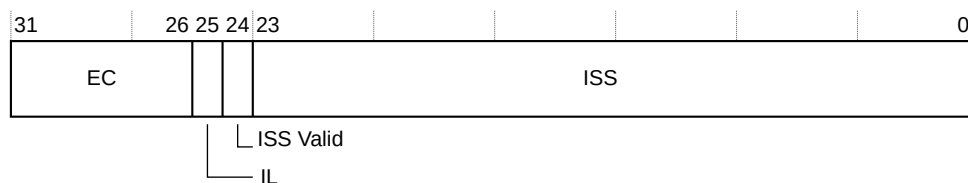
### 3.1.44 ESR\_EL1, Exception Syndrome Register, EL1

The ESR\_EL1 holds syndrome information for an exception taken to EL1.

#### Bit field descriptions

ESR\_EL1 is a 32-bit register, and is part of the Exception and fault handling registers functional group.

**Figure 3-25: ESR\_EL1 bit assignments**



#### EC, [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about.

#### IL, [25]

Instruction Length for synchronous exceptions. The possible values are:

0	16-bit.
1	32-bit.

This field is 1 for the SError interrupt, instruction aborts, misaligned PC, Stack pointer misalignment, Data Aborts for which the ISV bit is 0, exceptions caused by an illegal instruction set state, and exceptions using the 0x00 Exception Class.

#### ISS Valid, [24]

Syndrome valid. The possible values are:

0	ISS not valid, ISS is <b>RES0</b> .
1	ISS valid.

#### ISS, [23:0]

Syndrome information.

When the EC field is 0x2F, indicating an SError interrupt has occurred, the ISS field contents are **IMPLEMENTATION DEFINED**.

#### Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.45 ESR\_EL2, Exception Syndrome Register, EL2

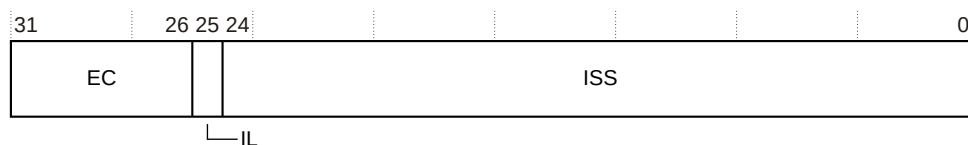
The ESR\_EL2 holds syndrome information for an exception taken to EL2.

#### Bit field descriptions

ESR\_EL2 is a 32-bit register, and is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.

**Figure 3-26: ESR\_EL2 bit assignments**



#### EC, [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information.

**IL, [25]**

Instruction Length for synchronous exceptions. The possible values are:

0	16-bit.
1	32-bit.

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

**ISS, [24:0]**

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

When reporting a virtual SEI, bits[24:0] take the value of VSESRL\_EL2[24:0].

When reporting a physical SEI, the following occurs:

- IDS==0 (architectural syndrome).
- AET always reports an uncontrollable error (UC) with value 0b000 or an unrecoverable error (UEU) with value 0b001.
- EA is **RES0**.

When reporting a synchronous Data Abort, EA is **RES0**.

See [3.1.82 VESR\\_EL2, Virtual SError Exception Syndrome Register](#) on page 188.

**Configurations**

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

RW fields in this register reset to architecturally **UNKNOWN** values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.46 ESR\_EL3, Exception Syndrome Register, EL3**

The ESR\_EL3 holds syndrome information for an exception taken to EL3.

**Bit field descriptions**

ESR\_EL3 is a 32-bit register, and is part of the Exception and fault handling registers functional group.

**Figure 3-27: ESR\_EL3 bit assignments****EC, [31:26]**

Exception Class. Indicates the reason for the exception that this register holds information about.

**IL, [25]**

Instruction Length for synchronous exceptions. The possible values are:

0	16-bit.
1	32-bit.

This field is 1 for the SError interrupt, instruction aborts, misaligned PC, Stack pointer misalignment, data aborts for which the ISV bit is 0, exceptions caused by an illegal instruction set state, and exceptions using the 0x0 Exception Class.

**ISS Valid, [24]**

Syndrome valid. The possible values are:

0	ISS not valid, ISS is <b>RES0</b> .
1	ISS valid.

**ISS, [23:0]**

Syndrome information.

When the EC field is 0x2F, indicating an SError interrupt has occurred, the ISS field contents are **IMPLEMENTATION DEFINED**.

**Configurations**

RW fields in this register reset to architecturally **UNKNOWN** values.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.47 HACR\_EL2, Hyp Auxiliary Configuration Register, EL2**

HACR\_EL2 controls trapping to EL2 of **IMPLEMENTATION DEFINED** aspects of Non-secure EL1 or EL0 operation. This register is not used in the Cortex®-A65AE core.

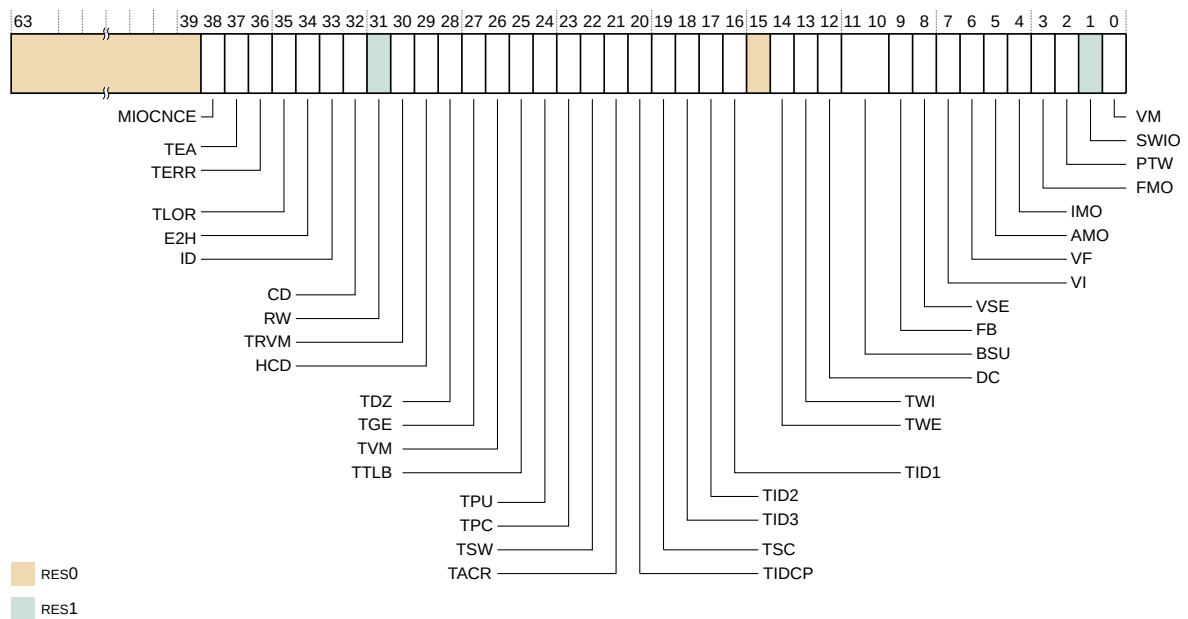
### 3.1.48 HCR\_EL2, Hypervisor Configuration Register, EL2

The HCR\_EL2 provides configuration control for virtualization, including whether various Non-secure operations are trapped to EL2.

#### Bit field descriptions

HCR\_EL2 is a 64-bit register, and is part of the Virtualization registers functional group.

**Figure 3-28: HCR\_EL2 bit assignments**



#### RES0, [63:39]

RES0 Reserved.

#### MIOCNCNCE, [38]

Mismatched Inner/Outer Cacheable Non-Coherency Enable, for the Non-secure EL1 and ELO translation regime.

#### RW, [31]

RES1 Reserved.

#### TGE, [27]

Traps general exceptions. If this bit is set, and SCR\_EL3.NS is set, then:

- All exceptions that would be routed to EL1 are routed to EL2.
- The SCTLR\_EL1.M bit is treated as 0 regardless of its actual state, other than for reading the bit.

- The HCR\_EL2.FMO, IMO, and AMO bits are treated as 1 regardless of their actual state, other than for reading the bits.
- All virtual interrupts are disabled.
- Any **IMPLEMENTATION DEFINED** mechanisms for signaling virtual interrupts are disabled.
- An exception return to EL1 is treated as an illegal exception return.

HCR\_EL2.TGE must not be cached in a TLB.

When the value of SCR\_EL3.NS is 0, the core behaves as if this field is 0 for all purposes other than a direct read or Write-Access of HCR\_EL2.

### TID3, [18]

Traps ID group 3 registers. The possible values are:

0	ID group 3 register accesses are not trapped.
1	Reads to ID group 3 registers that are executed from Non-secure EL1 are trapped to EL2.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for the registers covered by this setting.

### RES0, [15]

RES0	Reserved.
------	-----------

### Configurations

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

RW fields in this register reset to architecturally **UNKNOWN** values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 3.1.49 ID\_AA64AFR0\_EL1, AArch64 Auxiliary Feature Register 0

The core does not use this register, ID\_AA64AFR0\_EL1 is **RES0**.

## 3.1.50 ID\_AA64AFR1\_EL1, AArch64 Auxiliary Feature Register 1

The core does not use this register, ID\_AA64AFR0\_EL1 is **RES0**.

### 3.1.51 ID\_AA64DFR0\_EL1, AArch64 Debug Feature Register 0, EL1

Provides top-level information about the debug system in AArch64.

#### Bit field descriptions

ID\_AA64DFR0\_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

#### Figure 3-29: ID\_AA64DFR0\_EL1 bit assignments

##### RES0, [63:32]

**RES0**

Reserved.

##### CTX\_CMPs, [31:28]

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

**0x1**

Two breakpoints are context-aware.

##### RES0, [27:24]

**RES0**

Reserved.

##### WRPs, [23:20]

The number of watchpoints minus 1:

**0x3**

Four watchpoints.

##### RES0, [19:16]

**RES0**

Reserved.

##### BRPs, [15:12]

The number of breakpoints minus 1:

**0x5**

Six breakpoints.

##### PMUVer, [11:8]

Performance Monitors Extension version.

**0x4**

Performance monitor System registers implemented, PMUv3.

**TraceVer, [7:4]**

Trace extension:

**0x0**

Trace System registers not implemented.

**DebugVer, [3:0]**

Debug architecture version:

**0x8**

Arm®v8-A debug architecture implemented.

**Configurations**

ID\_AA64DFR0\_EL1 is architecturally mapped to external register EDDFR.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.52 ID\_AA64DFR1\_EL1, AArch64 Debug Feature Register 1, EL1**

This register is reserved for future expansion of top-level information about the debug system in AArch64 state.

**3.1.53 ID\_AA64ISAR0\_EL1, AArch64 Instruction Set Attribute Register 0, EL1**

The ID\_AA64ISAR0\_EL1 provides information about the instructions that are implemented in AArch64 state, including the instructions that are provided by the Cryptographic Extension.

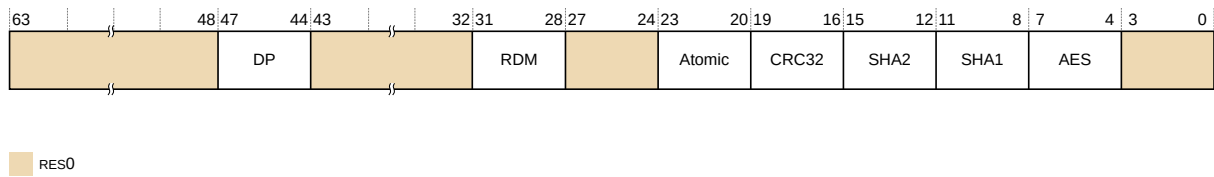
**Bit field descriptions**

ID\_AA64ISAR0\_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

The optional Cryptographic Extension is not included in the base product of the core. Arm requires licensees to have contractual rights to obtain the Cryptographic Extension.



**Figure 3-30: ID\_AA64ISAR0\_EL1 bit assignments****RES0, [63:48]**

RES0 Reserved.

**DP, [47:44]**

Indicates whether Dot Product support instructions are implemented.

0x1 UDOT, SDOT instructions are implemented.

**RES0, [43:32]**

RES0 Reserved.

**RDM, [31:28]**

Indicates whether SQRDMLAH and SQRDMLSH instructions in AArch64 are implemented.

0x1 SQRDMLAH and SQRDMLSH instructions are implemented.

**RES0, [27:24]**

RES0 Reserved.

**Atomic, [23:20]**

Indicates whether Atomic instructions in AArch64 are implemented. The value is:

0x2 LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions are implemented.

**CRC32, [19:16]**

Indicates whether CRC32 instructions are implemented. The value is:

0x1 CRC32 instructions are implemented.

**SHA2, [15:12]**

Indicates whether SHA2 instructions are implemented. The possible values are:

0x0 No SHA2 instructions are implemented. This is the value if the core implementation does not include the Cryptographic Extension.

0x1 SHA256H, SHA256H2, SHA256U0, and SHA256U1 implemented. This is the value if the core implementation includes the Cryptographic Extension.

**SHA1, [11:8]**

Indicates whether SHA1 instructions are implemented. The possible values are:

0x0	No SHA1 instructions implemented. This is the value if the core implementation does not include the Cryptographic Extension.
0x1	SHA1C, SHA1P, SHA1M, SHA1SU0, and SHA1SU1 implemented. This is the value if the core implementation includes the Cryptographic Extension.

**AES, [7:4]**

Indicates whether AES instructions are implemented. The possible values are:

0x0	No AES instructions implemented. This is the value if the core implementation does not include the Cryptographic Extension.
0x2	AESE, AESD, AESMC, and AESIMC implemented, plus PMULL and PMULL2 instructions operating on 64-bit data. This is the value if the core implementation includes the Cryptographic Extension.

**RES0, [3:0]**

RES0	Reserved.
------	-----------

**Configurations**

ID\_AA64ISAR0\_EL1 is architecturally mapped to external register ID\_AA64ISAR0.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

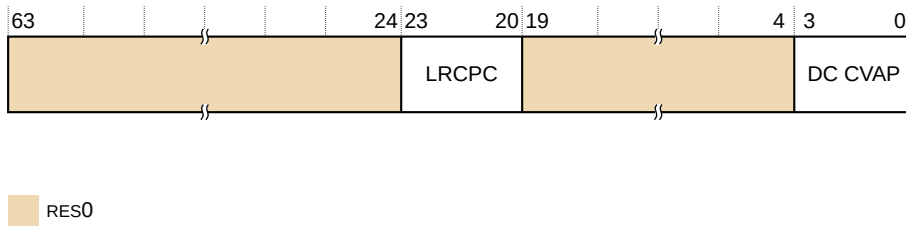
**3.1.54 ID\_AA64ISAR1\_EL1, AArch64 Instruction Set Attribute Register 1, EL1**

The ID\_AA64ISAR1\_EL1 provides information about the instructions that are implemented in AArch64 state.

**Bit field descriptions**

ID\_AA64ISAR1\_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

**Figure 3-31: ID\_AA64ISAR1\_EL1 bit assignments****RES0, [63:24]****RES0**

Reserved.

**LRCPC, [23:20]**

Indicates whether load-acquire (LDA) instructions are implemented for a Release Consistent core consistent RCPC model.

0x1      The LDAPRB, LDAPRH, and LDAPR instructions are implemented in AArch64.

**RES0, [19:4]****RES0**

Reserved.

**DC CVAP, [3:0]**

Indicates whether data cache, Clean to the Point of Persistence (DC CVAP) instructions are implemented.

0x1      DC CVAP is supported in AArch64.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.55 ID\_AA64MMFR0\_EL1, AArch64 Memory Model Feature Register 0, EL1

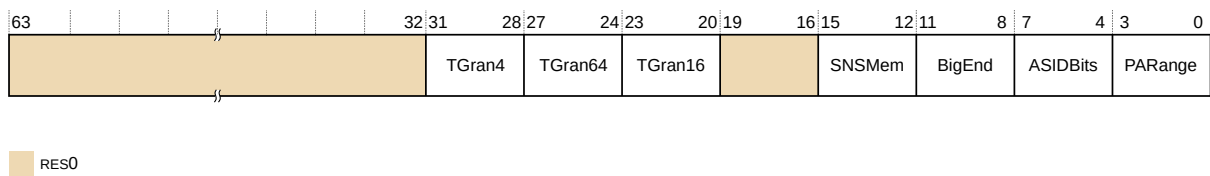
The ID\_AA64MMFR0\_EL1 provides information about the implemented memory model and memory management support in the AArch64 Execution state.

## Bit field descriptions

ID\_AA64MMFR0\_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

**Figure 3-32: ID\_AA64MMFR0\_EL1 bit assignments**



## RES0, [63:32]

**RES0**

Reserved.

## TGran4, [31:28]

Support for 4KB memory translation granule size:

0x0

4KB granule supported.

## TGran64, [27:24]

Support for 64KB memory translation granule size:

0x0

64KB granule supported.

## TGran16, [23:20]

Support for 16KB memory translation granule size:

## 0x1

Indicates that the 16KB granule is supported.

## RES0, [19:16]

## RES0

Reserved.

**SNSMem, [15:12]**

Secure versus Non-secure Memory distinction:

**0x1**

Supports a distinction between Secure and Non-secure Memory.

**BigEnd, [11:8]**

Mixed-endian configuration support:

**0x1**

Mixed-endian support. The SCTLR\_ELx.EE and SCTLR\_EL1.EOE bits can be configured.

**ASIDBits, [7:4]**

Number of ASID bits:

**0x2**

16 bits.

**PARange, [3:0]**

Physical address range supported:

**0x4**

44 bits, 16TB.

The supported Physical Address Range is 44 bits. Other cores in the DSU-AE might support a different Physical Address Range.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.56 ID\_AA64MMFR1\_EL1, AArch64 Memory Model Feature Register 1, EL1

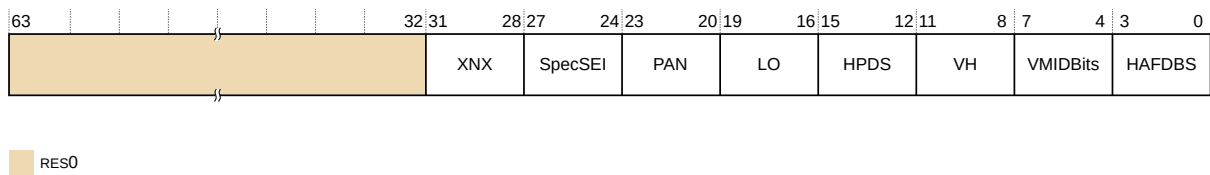
The ID\_AA64MMFR1\_EL1 provides information about the implemented memory model and memory management support in the AArch64 Execution state.

**Bit field descriptions**

ID\_AA64MMFR1\_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

**Figure 3-33: ID\_AA64MMFR1\_EL1 bit assignments**



## RES0, [63:32]

**RES0**

Reserved.

## XNX, [31:28]

Indicates whether provision of EL0 vs EL1 execute-never control at stage 2 is supported.

0x1

EL0/EL1 execute control distinction at stage 2 bit is supported. All other values are reserved.

## SpecSEI, [27:24]

Describes whether the PE can generate SError interrupt exceptions from Speculative reads of memory, including Speculative instruction fetches.

**0x0**

The PE never generates an SError interrupt due to an External abort on a Speculative read.

PAN, [23:20]

Privileged Access Never. Indicates support for the PAN bit in PSTATE, SPSR\_EL1, SPSR\_EL2, SPSR\_EL3, and DSPSR\_EL0.

## 0x2

PAN supported and AT S1E1RP and AT S1E1WP instructions supported.

LO, [19:16]

Indicates support for LORegions.

0x1

LORegions are supported.

## HPDS, [15:12]

Presence of Hierarchical Disables. Enables an operating system or hypervisor to hand over up to 4 bits of the last level translation table descriptor (bits[62:59] of the translation table entry) for use by hardware for **IMPLEMENTATION DEFINED** usage. The value is:

## 0x2

Hierarchical Permission Disables and Hardware allocation of bits[62:59] supported.  
PBHA is not implemented.

**VH, [11:8]**

Indicates whether Virtualization Host Extensions are supported.

**0x1**

Virtualization Host Extensions supported.

**VMIDBits, [7:4]**

Indicates the number of VMID bits supported.

**0x2**

16 bits are supported.

**HAFDBS, [3:0]**

Indicates the support for hardware updates to Access flag and dirty state in translation tables.

**0x2**

Hardware update of both the Access flag and dirty state is supported in hardware.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.57 ID\_AA64MMFR2\_EL1, AArch64 Memory Model Feature Register 2, EL1

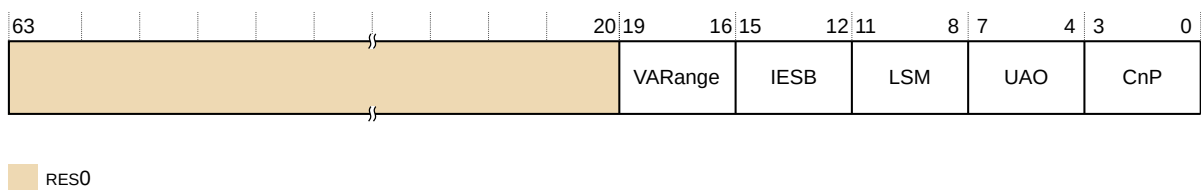
The ID\_AA64MMFR2\_EL1 provides information about the implemented memory model and memory management support in the AArch64 Execution state.

**Bit field descriptions**

ID\_AA64MMFR2\_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

**Figure 3-34: ID\_AA64MMFR2\_EL1 bit assignments**



**RES0, [63:20]****RES0**

Reserved.

**VARange, [19:16]**

Indicates support for a larger virtual address. The value is:

**0x0**

VMSAv8-64 supports 48-bit virtual addresses.

**IESB, [15:12]**

Indicates whether an implicit Error Synchronization Barrier has been inserted. The value is:

**0x1**SCTLR\_ELx.IESB implicit `ErrorSynchronizationBarrier` control implemented.**LSM, [11:8]**

Indicates whether STM ordering control bits are supported. The value is:

**0x0**

LSMAOE and nTLSMD bit not supported.

**UAO, [7:4]**Indicates the presence of the *User Access Override* (UAO). The value is:**0x1**

UAO is supported.

**CnP, [3:0]**

Common not Private. Indicates whether a TLB entry is pointed at a translation table base register that is a member of a common set. The value is:

**0x1**

CnP bit is supported.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.



### 3.1.58 ID\_AA64PFR0\_EL1, AArch64 Processor Feature Register 0, EL1

The ID\_AA64PFR0\_EL1 provides additional information about implemented core features in AArch64.

The optional Advanced SIMD and floating-point support is not included in the base product of the core. Arm requires licensees to have contractual rights to obtain the Advanced SIMD and floating-point support.

#### Bit field descriptions

ID\_AA64PFR0\_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

**Figure 3-35: ID\_AA64PFR0\_EL1 bit assignments**

63	60	59	56	55	32	31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0																		
CSV3				CSV2				RAS				GIC				AdvSIMD				FP				EL3 handling				EL2 handling				EL1 handling				EL0 handling			

 RES0

#### CSV3, [63:60]

**0x1**

Data that are loaded under speculation with a permission or domain fault cannot be used to form an address or generate condition codes to be used by instructions newer than the load in the speculative sequence. This is the reset value.

All other values reserved.

#### CSV2, [59:56]

**0x1**

Branch targets trained in one context cannot affect speculative execution in a different hardware described context. This is the reset value.

All other values reserved.

#### RES0, [55:32]

**RES0**

Reserved.

#### RAS, [31:28]

RAS extension version. The possible values are:

**0x1**

Version 1 of the RAS extension is present.

**GIC, [27:24]**

GIC CPU interface:

**0x0**

GIC CPU interface is disabled, GICCDISABLE is HIGH, or not implemented.

**0x1**

GIC CPU interface is implemented and enabled, GICCDISABLE is LOW. GICv4 is supported.

**AdvSIMD, [23:20]**

Advanced SIMD. The possible values are:

**0x1**

Advanced SIMD, including half-precision support, is implemented.

**0xF**

Advanced SIMD is not implemented.

The FP and AdvSIMD both take the same value, as both must be implemented, or neither.

**FP, [19:16]**

Floating-point. The possible values are:

**0x1**

Floating-point, including half-precision support, is implemented.

**0xF**

Floating-point is not implemented.

The FP and AdvSIMD both take the same value, as both must be implemented, or neither.

**EL3 handling, [15:12]**

EL3 exception handling:

**0x1**

Instructions can be executed at EL3 in AArch64 state only.

**EL2 handling, [11:8]**

EL2 exception handling:

**0x1**

Instructions can be executed at EL3 in AArch64 state only.

**EL1 handling, [7:4]**

EL1 exception handling. The possible values are:

**0x1**

Instructions can be executed at EL3 in AArch64 state only.

**ELO handling, [3:0]**

ELO exception handling. The possible values are:

**0x1**

Instructions can be executed at EL3 in AArch64 state only.

**Configurations**

ID\_AA64PFR0\_EL1 is architecturally mapped to External register EDPFR.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.59 ID\_AA64PFR1\_EL1, AArch64 Processor Feature Register 1, EL1**

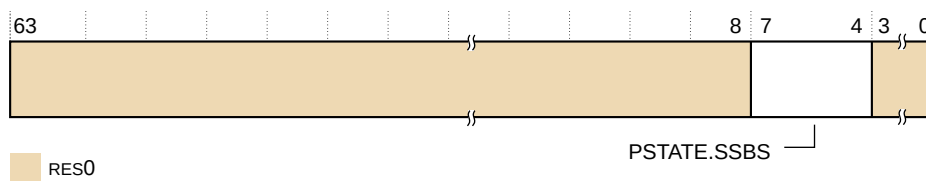
The ID\_AA64PFR1\_EL1 provides additional information about implemented core features in AArch64.

**Bit field descriptions**

ID\_AA64PFR1\_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is Read Only.

**Figure 3-36: ID\_AA64PFR1\_EL1 bit assignments**

**RES0, [63:8]**

**RES0** Reserved.

**SSBS, [7:4]**

AArch64 provides the PSTATE.SSBS mechanism to mark regions that are *Speculative Store Bypassing Safe* (SSBS).

**0x01** The MSR/MRS instructions are not implemented to directly read and write the PSTATE.SSBS field.

**RES0, [3:0]**

**RES0** Reserved.

**Configurations**

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

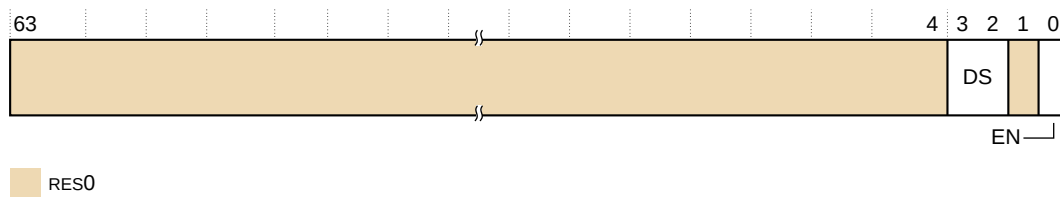
### 3.1.60 LORC\_EL1, LORegion Control Register, EL1

The LORC\_EL1 register enables and disables LORegions, and selects the current LORegion descriptor.

#### Bit field descriptions

LORC\_EL1 is a 64-bit register and is part of the Virtual memory control registers functional group.

**Figure 3-37: LORC\_EL1 bit assignments**



#### RES0, [63:4]

Reserved, **RES0**.

#### DS, [3:2]

Descriptor Select. Number that selects the current LORegion descriptor that is accessed by the LORSA\_EL1, LOREA\_EL1, and LORN\_EL1 registers.

#### RES0, [1]

Reserved, **RES0**.

#### EN, [0]

Enable. The possible values are:

- |   |                                    |
|---|------------------------------------|
| 0 | Disabled. This is the reset value. |
| 1 | Enabled.                           |

#### Configurations

RW fields in this register reset to architecturally **UNKNOWN** values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

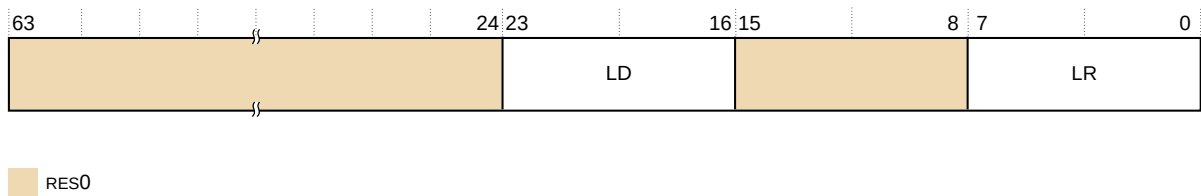
### 3.1.61 LORID\_EL1, LORegion ID Register, EL1

The LORID\_EL1 ID register indicates the supported number of LORegions and LORegion descriptors.

#### Bit field descriptions

LORID\_EL1 is a 64-bit register.

**Figure 3-38: LORID\_EL1 bit assignments**



#### RES0, [63:24]

Reserved, **RES0**.

#### LD, [23:16]

Number of LORegion descriptors supported by the implementation, expressed as binary 8-bit number. The value is:

0x04      Four LORegion descriptors are supported.

#### RES0, [15:8]

Reserved, **RES0**.

#### LR, [7:0]

Number of LORegions supported by the implementation, expressed as a binary 8-bit number. The value is:

0x04      Four LORegions are supported.

#### Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

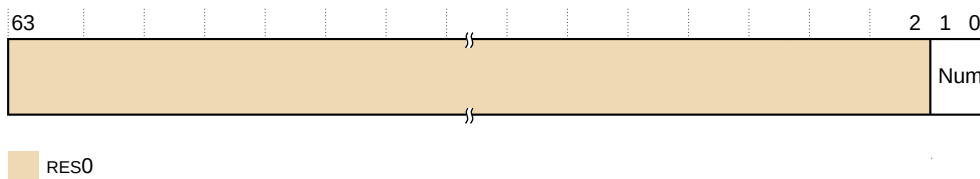
### 3.1.62 LORN\_EL1, LORegion Number Register, EL1

The LORN\_EL1 register holds the number of the LORegion described in the current LORegion descriptor that is selected by LORC\_EL1.DS.

#### Bit field descriptions

LORN\_EL1 is a 64-bit register and is part of the Virtual memory control registers functional group.

**Figure 3-39: LORN\_EL1 bit assignments**



#### RES0, [63:2]

Reserved, **RES0**.

#### Num, [1:0]

Indicates the LORegion number.

#### Configurations

RW fields in this register reset to architecturally **UNKNOWN** values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

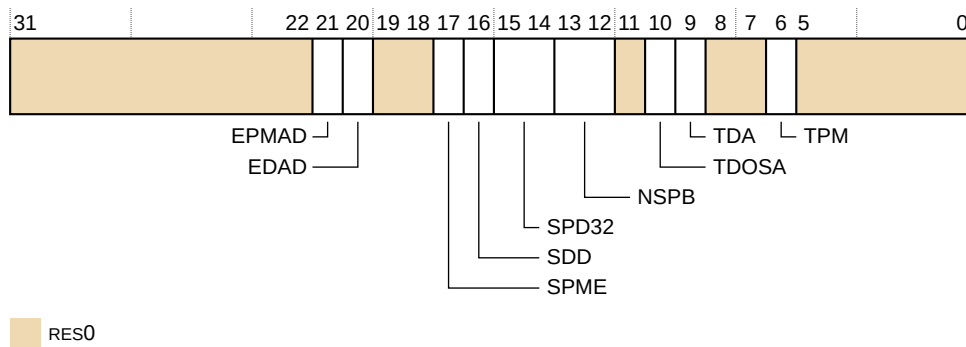
### 3.1.63 MDCR\_EL3, Monitor Debug Configuration Register, EL3

The MDCR\_EL3 provides configuration options for Security to self-hosted debug.

#### Bit field descriptions

MDCR\_EL3 is a 32-bit register, and is part of:

- The Debug registers functional group.
- The Security registers functional group.

**Figure 3-40: MDCR\_EL3 bit assignments****RES0, [31:22]**

**RES0**      Reserved.

**EPMAD, [21]**

External debugger access to Performance Monitors registers disabled. This disables access to these registers by an external debugger. The possible values are:

- |   |   |
|---|---|
| 0 | Access to Performance Monitors registers from external debugger is permitted.   |
| 1 | Access to Performance Monitors registers from external debugger is disabled, unless overridden by authentication interface. |

**EDAD, [20]**

External debugger access to breakpoint and watchpoint registers disabled. This disables access to these registers by an external debugger. The possible values are:

- |   |  |
|---|--|
| 0 | Access to breakpoint and watchpoint registers from external debugger is permitted.   |
| 1 | Access to breakpoint and watchpoint registers from external debugger is disabled, unless overridden by authentication interface. |

**SPME, [17]**

Secure performance monitors enable. This enables event counting exceptions from Secure state. The possible values are:

- |   |  |
|---|--|
| 0 | Event counting prohibited in Secure state. |
| 1 | Event counting allowed in Secure state.    |

**SPD32, [15:14]**

**RES0**  
Reserved.

**TDOSA, [10]**

Trap accesses to the OS Debug system registers, OSLAR\_EL1, OSLSR\_EL1, OSDLR\_EL1, and DBGPRCR\_EL1 OS.

- |   |   |
|---|---|
| 0 | Accesses are not trapped.                                     |
| 1 | Accesses to the OS Debug system registers are trapped to EL3. |

The reset value is **UNKNOWN**.

**TDA, [9]**

Trap accesses to the remaining sets of Debug registers to EL3.

- |   |  |
|---|--|
| 0 | Accesses are not trapped.  |
| 1 | Accesses to the remaining Debug system registers are trapped to EL3. |

The reset value is **UNKNOWN**.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.64 MIDR\_EL1, Main ID Register, EL1**

The MIDR\_EL1 provides identification information for the core, including an implementer code for the device and a device ID number.

**Bit field descriptions**

MIDR\_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

**Figure 3-41: MIDR\_EL1 bit assignments**

31	24 23	20 19	16 15	4	3	0
Implementer	Variant	Architecture	PartNum	Revision		

**Implementer, [31:24]**

Indicates the implementer code. This value is:

- |      |  |
|------|--|
| 0x41 | ASCII character 'A' - implementer is Arm® Limited. |
|------|--|

**Variant, [23:20]**

Indicates the variant number of the core. This is the major revision number x in the rx part of the rxy description of the product revision status. This value is:



0x1            r1p1.

Architecture, [19:16]

Indicates the architecture code. This value is:

0xF            Defined by CPUID scheme.

PartNum, [15:4]

Indicates the primary part number. This value is:

0xD43        Cortex®-A65AE core.

Revision, [3:0]

Indicates the minor revision number of the core. This is the minor revision number y in the py part of the rxy description of the product revision status. This value is:

0x1            r1p1.

Configurations

The MIDR\_EL1 is architecturally mapped to external MIDR\_EL1 register.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

3.1.65 MPIDR\_EL1, Multiprocessor Affinity Register, EL1

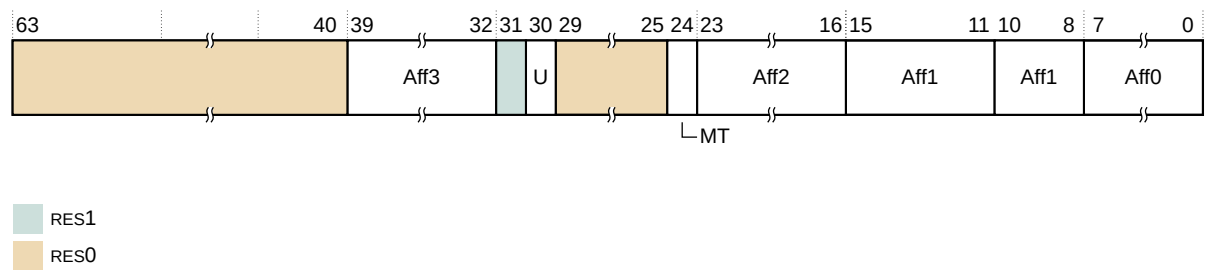
The MPIDR\_EL1 provides an additional core identification mechanism for scheduling purposes in a cluster.

Bit field descriptions

MPIDR\_EL1 is a 64-bit register, and is part of the Other system control registers functional group.

This register is read-only.

Figure 3-42: MPIDR\_EL1 bit assignments



**RES0, [63:40]**

<b>RES0</b>	Reserved.
-------------	-----------

**Aff3, [39:32]**

Affinity level 3. Highest level affinity field.

**CLUSTERID**

Indicates the value read in the **CLUSTERIDAFF3** configuration signal.

**RES1, [31]**

<b>RES1</b>	Reserved.
-------------	-----------

**U, [30]**

Indicates a single core system, as distinct from core 0 in a cluster. This value is:

0	Core is part of a multiprocessor system. This is the value for implementations with more than one core, and for implementations with an ACE or CHI master interface.
---	--

**RES0, [29:25]**

<b>RES0</b>	Reserved.
-------------	-----------

**MT, [24]**

Indicates whether the lowest level of affinity consists of logical cores that are implemented using a multithreading type approach. This value is:

1	Performance of PEs at the lowest affinity level is interdependent.
---	--

Affinity0 represents threads, Cortex®-A65AE is multithreaded.

**Aff2, [23:16]**

Affinity L2. Second highest level affinity field.

**CLUSTERID**

Indicates the value read in the **CLUSTERIDAFF2** configuration signal.

**Aff1, [15:11]**

Part of Affinity L1. Third highest level affinity field.

RAZ	Read-As-Zero.
-----	---------------

**Aff1, [10:8]**

Part of Affinity L1. Third highest level affinity field.

CPUID	Identification number for each CPU in the cluster in Split-mode.
0x0	MP1: CPUID: 0.

0x7	MP8: CPUID: 7.
CPUID	Identification number for each CPU in the cluster in Lock-mode.
0x0	MP1: CPUID: 0.
0x7	MP4: CPUID: 3.

**Aff0, [7:0]**  
Affinity level 0. The level identifies individual threads within a multithreaded core. The Cortex®-A65AE core is dual-threaded, so this field has the value:

0x00	Thread 0.
0x01	Thread 1.

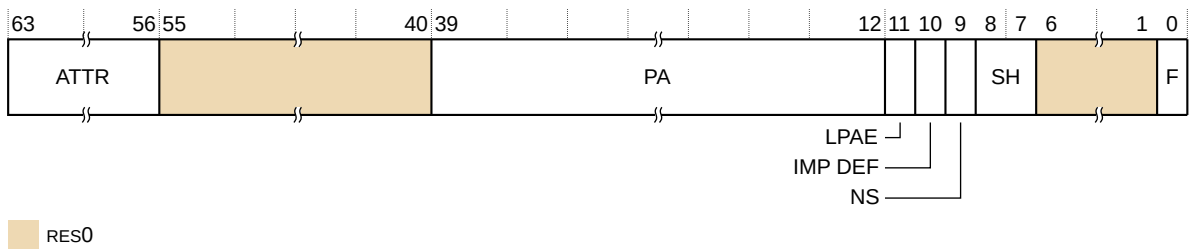
**Configurations**  
MPIDR\_EL1[31:0] is mapped to external register EDDEVAFF0.  
MPIDR\_EL1[63:32] is mapped to external register EDDEVAFF1.  
Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

3.1.66 PAR\_EL1, Physical Address Register, EL1

The PAR\_EL1 returns the output address from an address translation instruction that executed successfully, or fault information if the instruction did not execute successfully.

**Bit field descriptions, PAR\_EL1.F is 0**  
The following figure shows the PAR bit assignments when PAR.F is 0.

Figure 3-43: PAR bit assignments, PAR\_EL1.F is 0



**IMP DEF, [10]**  
IMPLEMENTATION DEFINED. Bit[10] is RES0.

**F, [0]**  
Indicates whether the instruction performed a successful address translation.

0	Address translation completed successfully.
1	Address translation aborted.

### Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### Bit field descriptions, PAR\_EL1.F is 1

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

## 3.1.67 REVIDR\_EL1, Revision ID Register, EL1

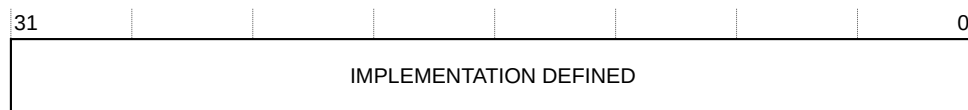
The REVIDR\_EL1 provides revision information, additional to MIDR\_EL1, that identifies minor fixes (errata) which might be present in a specific implementation of the Cortex®-A65AE core.

### Bit field descriptions

REVIDR\_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

**Figure 3-44: REVIDR\_EL1 bit assignments**



IMPLEMENTATION DEFINED, [31:0]

IMPLEMENTATION DEFINED.

### Configurations

There are no configuration notes.

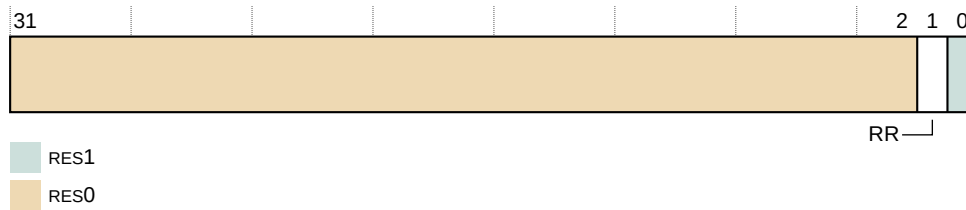
Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 3.1.68 RMR\_EL3, Reset Management Register

The RMR\_EL3 controls the Execution state that the core boots into and allows request of a Warm reset.

### Bit field descriptions

RMR\_EL3 is a 32-bit register, and is part of the Reset management registers functional group.

**Figure 3-45: RMR\_EL3 bit assignments****RES0, [31:2]**

RES0 Reserved.

**RR, [1]**

Reset Request. The possible values are:

- |   |  |
|---|--|
| 0 | This is the reset value on both a Warm and a Cold reset. |
| 1 | Requests a Warm reset.                                   |

The bit is strictly a request.

**RES1, [0]**

RES1 Reserved.

**Configurations**

There are no configuration notes.

Details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.69 RVBAR\_EL3, Reset Vector Base Address Register, EL3**

RVBAR\_EL3 contains the **IMPLEMENTATION DEFINED** address that execution starts from after reset.

**Bit field descriptions**

RVBAR\_EL3 is a 64-bit register, and is part of the Reset management registers functional group.

This register is read-only.

**Figure 3-46: RVBAR\_EL3 bit assignments**

**RVBA, [63:0]**

Reset Vector Base Address. The address that execution starts from after reset. Bits[1:0] of this register are 0b00, as this address must be aligned, and bits [63:48] are 0x0000 because the address must be within the physical address size that is supported by the core.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.70 SCTL\_EL1, System Control Register, EL1**

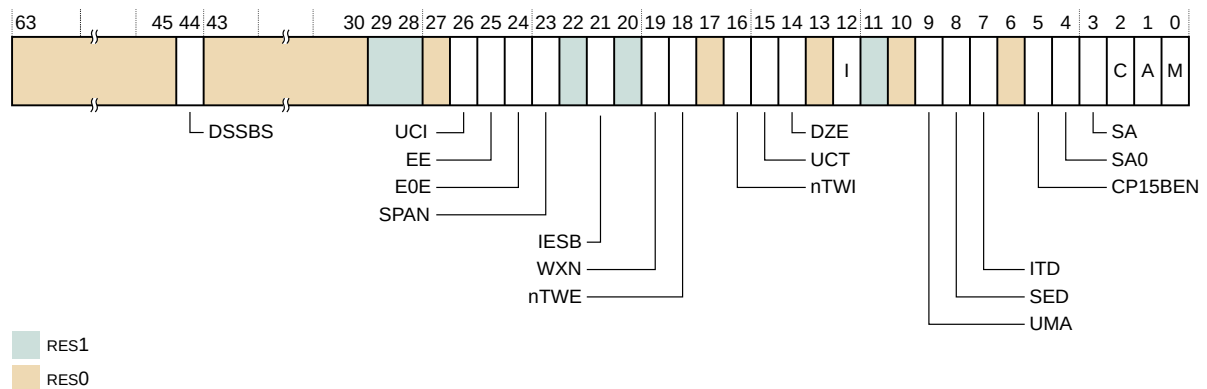
The SCTL\_EL1 provides top-level control of the system, including its memory system, at EL1 and EL0.

**Bit field descriptions**

SCTL\_EL1 is a 64-bit register, and is part of the Other system control registers functional group.

This register resets to 0x0000000030D50838.

**Figure 3-47: SCTL\_EL1 bit assignments**

**RES0, [63:45]**

RES0 Reserved

**DSSBS, [44]**

DSSBS is used to set the new PSTATE bit, SSBS (Speculative Store Bypassing Safe).

- |   |   |
|---|---|
| 0 | PSTATE.SSBS is set to 0 on an exception taken to this Exception level. This is the reset value. |
| 1 | PSTATE.SSBS is set to 1 on an exception taken to this Exception level.                          |

**RES0, [43:30]**

RES0	Reserved
------	----------

**RES1, [29:28]**

RES1	Reserved
------	----------

**RES0, [27]**

RES0	Reserved
------	----------

**EE, [25]**

Exception endianness. The value of this bit controls the endianness for explicit data accesses at EL1. This value also indicates the endianness of the translation table data for translation table lookups. The possible values of this bit are:

0	Little-endian.
1	Big-endian.

**ITD, [7]**

This field is RAZ/WI.

**RES0, [6]**

RES0	Reserved
------	----------

**CP15BEN, [5]**

CP15 barrier enable. The possible values are:

0	CP15 barrier operations disabled. Their encodings are <i>UNDEFINED</i> .
1	CP15 barrier operations enabled.

**M, [0]**

MMU enable. The possible values are:

0	EL1 and EL0 stage 1 MMU disabled.
1	EL1 and EL0 stage 1 MMU enabled.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.71 SCTLR\_EL2, System Control Register, EL2

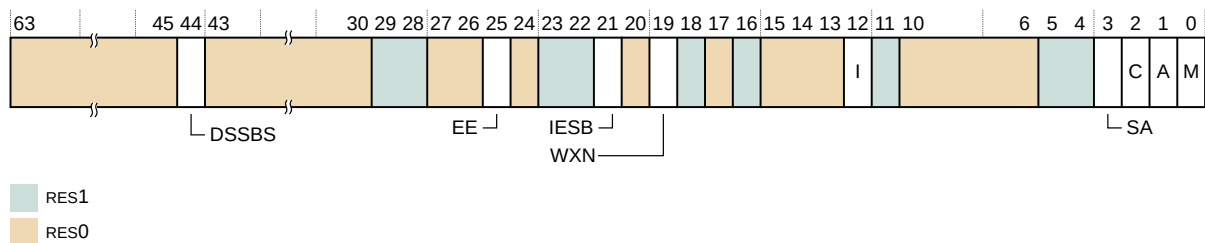
The SCTLR\_EL2 provides top-level control of the system, including its memory system at EL2.

#### Bit field descriptions

SCTLR\_EL2 is a 64-bit register, and is part of:

- The Virtualization registers functional group
- The Other system control registers functional group

**Figure 3-48: SCTLR\_EL2 bit assignments**



This register resets to 0x30C50838.

#### DSSBS, [44]

DSSBS is used to set the new PSTATE bit, SSBS (Speculative Store Bypassing Safe).

SCTLR\_EL2.DSSBS is held in bit[44] regardless of the value of HCR\_EL2.E2H or HCR\_EL2.TGE.

- |   |   |
|---|---|
| 0 | PSTATE.SSBS is set to 0 on an exception taken to this Exception level. This is the reset value. |
| 1 | PSTATE.SSBS is set to 1 on an exception taken to this Exception level.                          |

#### Configurations

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

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.72 SCTLR\_EL3, System Control Register, EL3

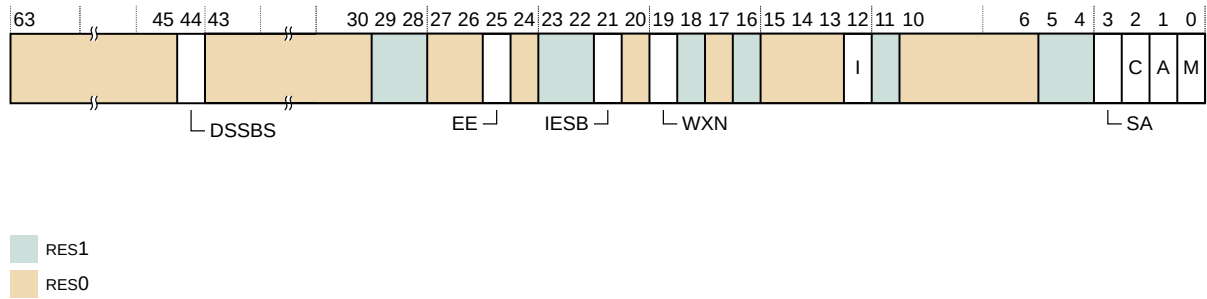
The SCTLR\_EL3 provides top-level control of the system, including its memory system at EL3.

#### Bit field descriptions

SCTLR\_EL3 is a 64-bit register, and is part of the Other system control registers functional group.

This register resets to 0x30C50838.



**Figure 3-49: SCTLR\_EL3 bit assignments****RES0, [63:45]**

RES0 Reserved

**DSSBS, [44]**

DSSBS is used to set the new PSTATE bit, SSBS (Speculative Store Bypassing Safe).

- |   |   |
|---|---|
| 0 | PSTATE.SSBS is set to 0 on an exception taken to this Exception level. This is the reset value. |
| 1 | PSTATE.SSBS is set to 1 on an exception taken to this Exception level.                          |

**RES0, [43:30]**

RES0 Reserved

**RES1, [29:28]**

RES1 Reserved

**RES0, [27:26]**

RES0 Reserved

**EE, [25]**

Exception endianness. This bit controls the endianness for:

- Explicit data accesses at EL3.
- Stage 1 translation table walks at EL3.

The possible values are:

- |   |               |
|---|---------------|
| 0 | Little-endian |
| 1 | Big-endian    |

The reset value is determined by the CFGEND configuration signal.

**I, [12]**

Global instruction cache enable. The possible values are:

0	Instruction caches disabled. This is the reset value.
1	Instruction caches enabled.

**C, [2]**

Global enable for data and unified caches. The possible values are:

0	Disables data and unified caches. This is the reset value.
1	Enables data and unified caches.

**M, [0]**

Global enable for the EL3 MMU. The possible values are:

0	Disables EL3 MMU. This is the reset value.
1	Enables EL3 MMU.

**Configurations**

There are no configuration notes.

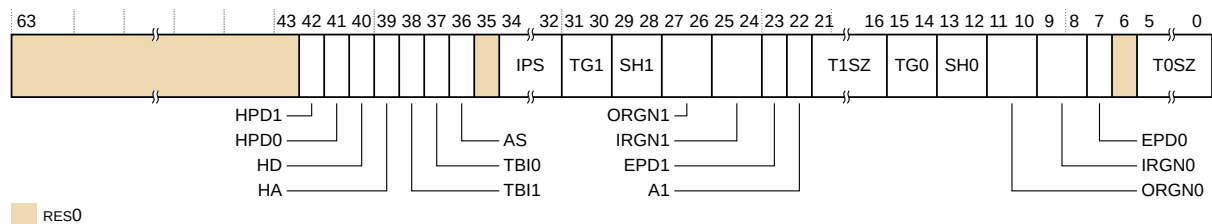
Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.73 TCR\_EL1, Translation Control Register, EL1**

The TCR\_EL1 determines which Translation base registers define the base address register for a translation table walk required for stage 1 translation of a memory access from EL0 or EL1 and holds Cacheability and Shareability information.

**Bit field descriptions**

TCR\_EL1 is a 64-bit register, and is part of the Virtual memory control registers functional group.

**Figure 3-50: TCR\_EL1 bit assignments****HD, [40]**

Hardware management of dirty state in stage 1 translations from EL0 and EL1. The possible values are:

0	Stage 1 hardware management of dirty state disabled.
1	Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1.

**HA, [39]**

Hardware Access flag update in stage 1 translations from EL0 and EL1. The possible values are:

0	Stage 1 Access flag update disabled.
1	Stage 1 Access flag update enabled.

**IPS, [34:32]**

Physical address size. The possible values are:

**0b000**

32 bits, 4GB.

**0b001**

36 bits, 64GB.

**0b010**

40 bits, 1TB.

**0b011**

42 bits, 4TB.

**0b100**

44 bits, 16TB.

Other values are reserved.

**Configurations**

RW fields in this register reset to **UNKNOWN** values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.74 TCR\_EL2, Translation Control Register, EL2**

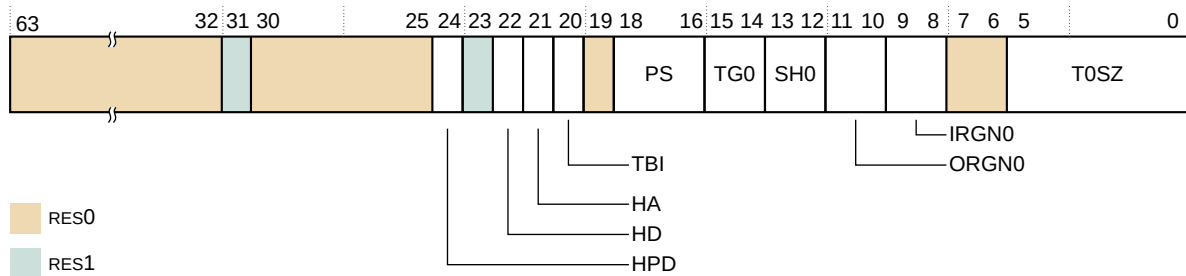
The TCR\_EL2 controls translation table walks required for stage 1 translation of a memory access from EL2 and holds Cacheability and Shareability information.

**Bit field descriptions**

TCR\_EL2 is a 64-bit register.

TCR\_EL2 is part of:

- The Virtual memory control registers functional group.
- The Hypervisor and virtualization registers functional group.

**Figure 3-51: TCR\_EL2 bit assignments****HD, [22]**

Dirty bit update. The possible values are:

- |   |                               |
|---|-------------------------------|
| 0 | Dirty bit update is disabled. |
| 1 | Dirty bit update is enabled.  |

**HA, [21]**

Stage 1 Access flag update. The possible values are:

- |   |   |
|---|---|
| 0 | Stage 1 Access flag update is disabled. |
| 1 | Stage 1 Access flag update is enabled.  |

**PS, [18:16]**

Physical address size. The possible values are:

- |       |                |
|-------|----------------|
| 0b000 | 32 bits, 4GB.  |
| 0b001 | 36 bits, 64GB. |
| 0b010 | 40 bits, 1TB.  |
| 0b011 | 42 bits, 4TB.  |
| 0b100 | 44 bits, 16TB. |

Other values are reserved.

**Configurations**

When the Virtualization Host Extension is activated, TCR\_EL2 has the same bit assignments as TCR\_EL1.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

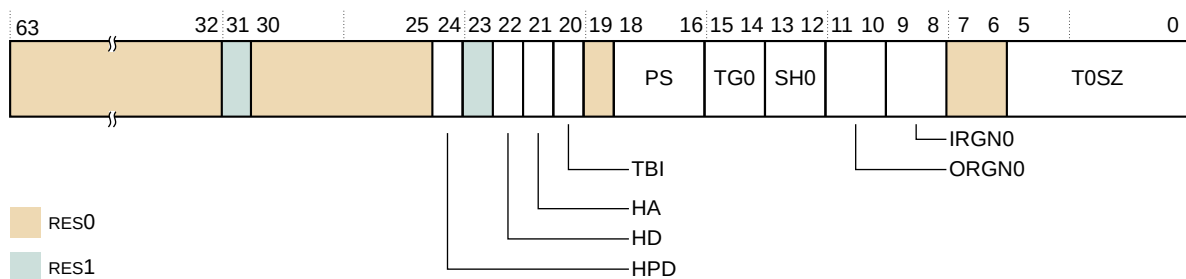
### 3.1.75 TCR\_EL3, Translation Control Register, EL3

The TCR\_EL3 controls translation table walks required for stage 1 translation of memory accesses from EL3 and holds cacheability and shareability information for the accesses.

#### Bit field descriptions

TCR\_EL3 is a 64-bit register, and is part of the Virtual memory control registers functional group.

**Figure 3-52: TCR\_EL3 bit assignments**



#### HPD, [24]

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by TTBR0\_EL3. The possible values are:

- |   |  |
|---|--|
| 0 | Hierarchical Permissions are enabled.  |
| 1 | Hierarchical Permissions are disabled. |



Note

In this case bit[61] (APTable[0]) and bit[59] (PXNTable) of the next level descriptor attributes are required to be ignored by the PE, and are no longer reserved, allowing them to be used by software.

#### HD, [22]

Hardware management of dirty state in stage 1 translations from EL3. The possible values are:

- |   |  |
|---|--|
| 0 | Stage 1 hardware management of dirty state disabled.                                     |
| 1 | Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1. |

Implementation of this bit is OPTIONAL, and, if not implemented, this bit is **RES0**.

#### HA, [21]

Hardware Access flag update in stage 1 translations from EL3. The possible values are:

- |   |                                      |
|---|--------------------------------------|
| 0 | Stage 1 Access flag update disabled. |
|---|--------------------------------------|

1 Stage 1 Access flag update enabled.

### PS, [18:16]

Physical address size. The possible values are:

0b000	32 bits, 4GB.
0b001	36 bits, 64GB.
0b010	40 bits, 1TB.
0b011	42 bits, 4TB.
0b100	44 bits, 16TB.

Other values are reserved.

### TG0, [15:14]

TTBR0\_EL3 granule size. The possible values are:

0b00	4KB.
0b10	16KB.
0b01	64KB.
0b11	Reserved.

All other values are not supported.

### SH0, [13:12]

Shareability attribute for memory associated with translation table walks using TTBR0\_EL3.

The possible values are:

0b00	Non-shareable.
0b01	Reserved.
0b10	Outer shareable.
0b11	Inner shareable.

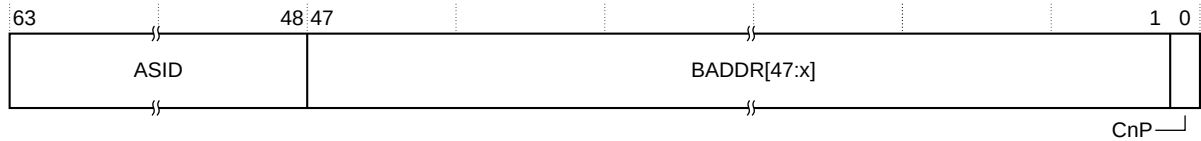
Bit fields and details not provided in this description are architecturally defined. See the Arm® *Architecture Reference Manual Armv8, for A-profile architecture*.

## 3.1.76 TTBR0\_EL1, Translation Table Base Register 0, EL1

The TTBR0\_EL1 holds the base address of translation table 0, and information about the memory it occupies. This is one of the translation tables for the stage 1 translation of memory accesses from modes other than Hyp mode.

### Bit field descriptions

TTBR0\_EL1 is 64-bit register.

**Figure 3-53: TTBR0\_EL1 bit assignments****ASID, [63:48]**

An ASID for the translation table base address. The TCR\_EL1.A1 field selects either TTBR0\_EL1.ASID or TTBR1\_EL1.ASID.

**BADDR[47:x], [47:1]**

Translation table base address, bits[47:x]. Bits [x-1:1] are *RES0*.

x is based on the value of TCR\_EL1.TOSZ, the stage of translation, and the memory translation granule size.

For instructions on how to calculate it, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The value of x determines the required alignment of the translation table, that must be aligned to  $2^x$  bytes.

If bits [x-1:1] are not all zero, this is a misaligned translation table base address. Its effects are **CONSTRAINED UNPREDICTABLE**, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value that is written.

**CnP, [0]**

Common not Private. The possible values are:

**0**

CnP is not supported.

**1**

CnP is supported.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

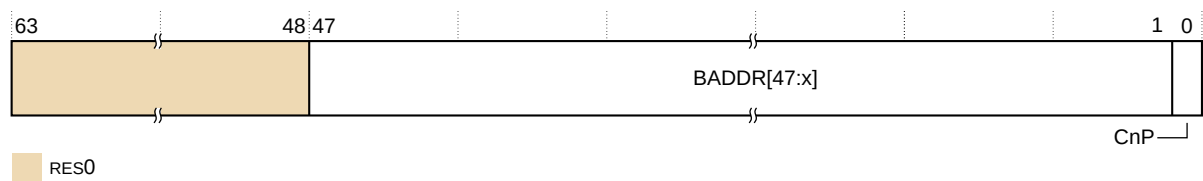
### 3.1.77 TTBR0\_EL2, Translation Table Base Register 0, EL2

The TTBR0\_EL2 holds the base address of the translation table for the stage 1 translation of memory accesses from EL2.

#### Bit field descriptions

TTBR0\_EL2 is a 64-bit register, and is part of the Virtual memory control registers functional group.

**Figure 3-54: TTBR0\_EL2 bit assignments**



#### RES0, [63:48]

##### RES0

Reserved.

#### BADDR, [47:1]

Translation table base address, bits[47:x]. Bits [x-1:1] are *RES0*.

x is based on the value of TCR\_EL2.TOSZ, the stage of translation, and the memory translation granule size.

For instructions on how to calculate it, see the *Arm® Architecture Reference Manual Arm®v8, for Arm®v8-A architecture profile*.

The value of x determines the required alignment of the translation table, that must be aligned to  $2^x$  bytes.

If bits [x-1:1] are not all zero, this is a misaligned translation table base address. Its effects are **CONSTRAINED UNPREDICTABLE**, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value that is written.

#### CnP, [0]

Common not Private. The possible values are:

**0**

CnP is not supported.

**1**

CnP is supported.



## Configurations

When the Virtualization Host Extension is activated, TTBR0\_EL2 has the same bit assignments as TTBR0\_EL1.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

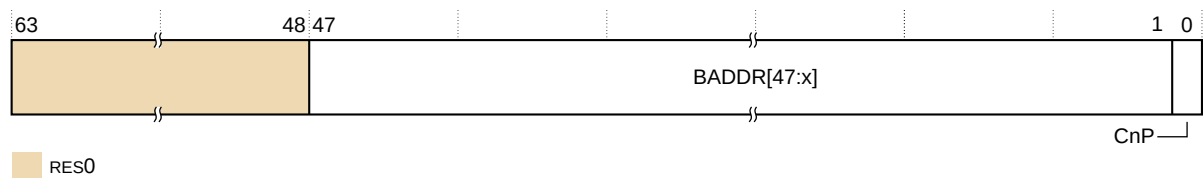
### 3.1.78 TTBR0\_EL3, Translation Table Base Register 0, EL3

The TTBR0\_EL3 holds the base address of the translation table for the stage 1 translation of memory accesses from EL3.

#### Bit field descriptions

TTBR0\_EL3 is a 64-bit register.

**Figure 3-55: TTBR0\_EL3 bit assignments**



#### [63:48]

Reserved, **RES0**.

#### BADDR[47:x], [47:1]

Translation table base address, bits[47:x]. Bits [x-1:1] are **RES0**.

x is based on the value of TCR\_EL1.TOSZ, the stage of translation, and the memory translation granule size.

For instructions on how to calculate it, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The value of x determines the required alignment of the translation table, that must be aligned to  $2^x$  bytes.

If bits [x-1:1] are not all zero, this is a misaligned translation table base address. Its effects are **CONSTRAINED UNPREDICTABLE**, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value that is written.

#### CnP, [0]

Common not Private. The possible values are:

0                      CnP is not supported.

1 CnP is supported.

## Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

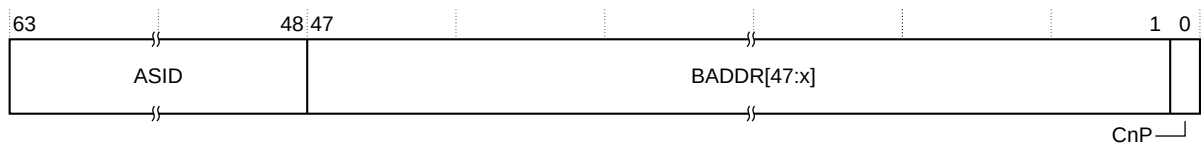
### 3.1.79 TTBR1\_EL1, Translation Table Base Register 1, EL1

The TTBR1\_EL1 holds the base address of translation table 1, and information about the memory it occupies. This is one of the translation tables for the stage 1 translation of memory accesses at EL0 and EL1.

## Bit field descriptions

TTBR1\_EL1 is a 64-bit register.

**Figure 3-56: TTBR1\_EL1 bit assignments**



## ASID, [63:48]

An ASID for the translation table base address. The TCR\_EL1.A1 field selects either TTBR0\_EL1.ASID or TTBR1\_EL1.ASID.

## BADDR[47:x], [47:1]

Translation table base address, bits[47:x]. Bits [x-1:0] are **RES0**.

x is based on the value of TCR\_EL1.TOSZ, the stage of translation, and the memory translation granule size.

For instructions on how to calculate it, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The value of x determines the required alignment of the translation table, that must be aligned to  $2^x$  bytes.

If bits [x-1:1] are not all zero, this is a misaligned Translation Table Base Address. Its effects are **CONSTRAINED UNPREDICTABLE**, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value that is written.

## CnP, [0]

Common not Private. The possible values are:

0	CnP is not supported.
1	CnP is supported.

### Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 3.1.80 TTBR1\_EL2, Translation Table Base Register 1, EL2

TTBR1\_EL2 has the same format and contents as TTBR1\_EL1.

See [3.1.79 TTBR1\\_EL1, Translation Table Base Register 1, EL1](#) on page 186.

## 3.1.81 VDISR\_EL2, Virtual Deferred Interrupt Status Register, EL2

The VDISR\_EL2 records that a virtual SError interrupt has been consumed by an `ESB` instruction executed at Non-secure EL1.

### Bit field descriptions

VDISR\_EL2 is a 64-bit register, and is part of the *Reliability, Availability, Serviceability* (RAS) registers functional group.

### Configurations

See [3.1.81.1 VDISR\\_EL2 at EL1 using AArch64](#) on page 187.

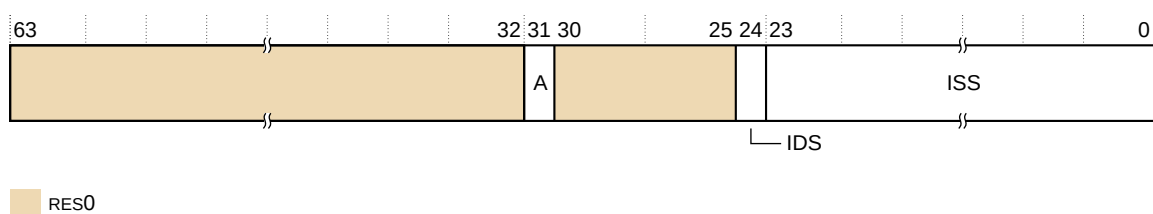
Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 3.1.81.1 VDISR\_EL2 at EL1 using AArch64

VDISR\_EL2 has a specific format when written at EL1.

The following figure shows the VDISR\_EL2 bit assignments when written at EL1 using AArch64:

**Figure 3-57: VDISR\_EL2 at EL1 using AArch64**



**RES0, [63:32]**

**RES0**      Reserved.

**A, [31]**

Set to 1 when ESB defers an asynchronous SError interrupt.

**RES0, [30:25]**

**RES0**      Reserved.

**IDS, [24]**

Contains the value from VESR\_EL2.IDS.

**ISS, [23:0]**

Contains the value from VESR\_EL2, bits[23:0].

### 3.1.82 VESR\_EL2, Virtual SError Exception Syndrome Register

The VESR\_EL2 provides the syndrome value that is reported to software on taking a virtual SError interrupt exception.

#### Bit field descriptions

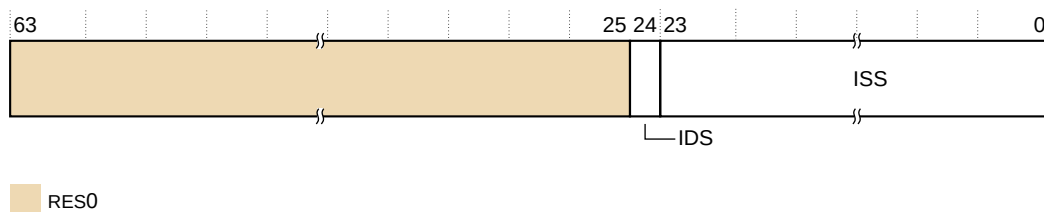
VESR\_EL2 is a 64-bit register, and is part of:

- The Exception and fault handling registers functional group.
- The Virtualization registers functional group.

If the virtual SError interrupt is taken to EL1, VESR\_EL2 provides the syndrome value that is reported in ESR\_EL1.

#### VESR\_EL2 bit assignments

**Figure 3-58: VESR\_EL2 bit assignments**

**RES0, [63:25]**

**RES0**      Reserved.

**IDS, [24]**

Indicates whether the deferred SError interrupt was of an **IMPLEMENTATION DEFINED** type. See ESR\_EL1.IDS for a description of the functionality.

On taking a virtual SError interrupt to EL1 using AArch64 because HCR\_EL2.VSE == 1, ESR\_EL1[24] is set to VSESR\_EL2.IDS.

**ISS, [23:0]**

Syndrome information. See ESR\_EL1.ISS for a description of the functionality.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**3.1.83 VTCR\_EL2, Virtualization Translation Control Register, EL2**

The VTCR\_EL2 controls the translation table walks required for the stage 2 translation of memory accesses from Non-secure EL0 and EL1.

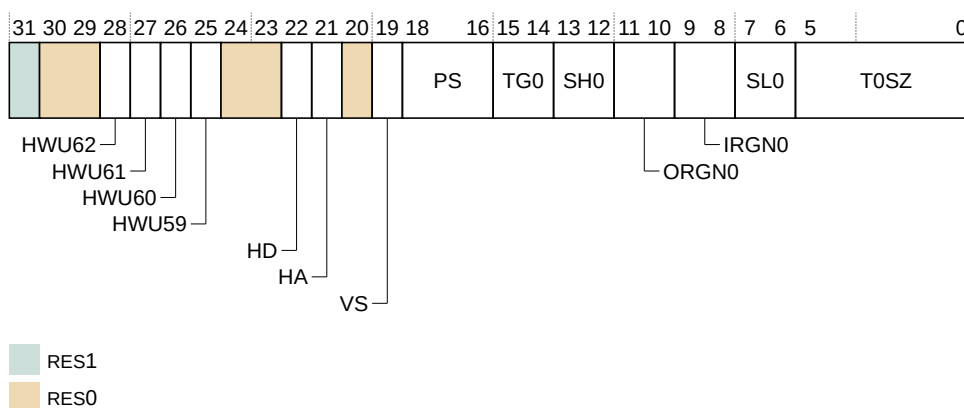
It also holds Cacheability and Shareability information for the accesses.

**Bit field descriptions**

VTCR\_EL2 is a 32-bit register, and is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

**Figure 3-59: VTCR\_EL2 bit assignments**





Bits[28:25] and bits[22:21], architecturally defined, are implemented in the core.

## TG0, [15:14]

TTBR0\_EL2 granule size. The possible values are:

00	4KB.
01	64KB.
10	16KB.
11	Reserved.

All other values are not supported.

## Configurations

RW fields in this register reset to architecturally **UNKNOWN** values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

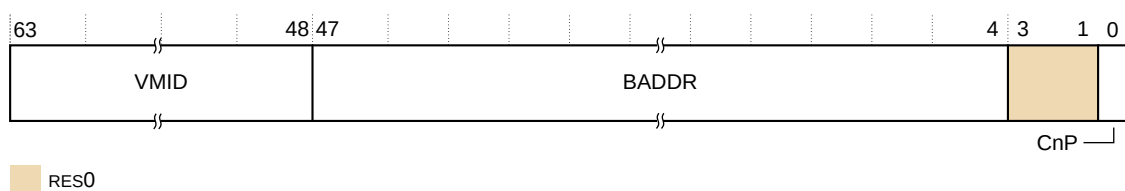
### 3.1.84 VTTBR\_EL2, Virtualization Translation Table Base Register, EL2

VTTBR\_EL2 holds the base address of the translation table for the stage 2 translation of memory accesses from Non-secure EL0 and EL1.

## Bit field descriptions

VTTBR\_EL2 is a 64-bit register.

**Figure 3-60: VTTBR\_EL2 bit assignments**



CnP, [0]

Common not Private. The possible values are:

0	CnP is not supported.
1	CnP is supported.

## Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 3.2 Error System registers

This chapter describes the error registers accessed by the AArch64 error registers.

### 3.2.1 Error System register summary

This section identifies the ERRO\* core error record registers accessed by the AArch64 ERX\* error registers.

For those registers that are not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The following table describes the architectural error record registers.

**Table 3-38: Architectural error System register summary**

Register mnemonic	Size	Register name	Access aliases from AArch64
ERROCTL	64	3.2.2 ERROCTL, Error Record Control Register on page 191	3.1.37 ERXCTL_EL1, Selected Error Record Control Register, EL1 on page 140
ERROFR	64	3.2.3 ERROFR, Error Record Feature Register on page 193	3.1.38 ERXFR_EL1, Selected Error Record Feature Register, EL1 on page 140
ERROMISCO	64	3.2.4 ERROMISCO, Error Record Miscellaneous Register 0 on page 195	3.1.39 ERXMISCO_EL1, Selected Error Record Miscellaneous Register 0, EL1 on page 141
ERROSTATUS	32	3.2.8 ERROSTATUS, Error Record Primary Status Register on page 202	3.1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1 on page 145

The following table describes the error record registers that are **IMPLEMENTATION DEFINED**.

**Table 3-39: IMPLEMENTATION DEFINED error System register summary**

Register mnemonic	Size	Register name	Access aliases from AArch64
ERROPFGCDN	32	3.2.5 ERROPFGCDN, Error Pseudo Fault Generation Count Down Register on page 198	3.1.40 ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1 on page 141
ERROPFGCTL	32	3.2.6 ERROPFGCTL, Error Pseudo Fault Generation Control Register on page 198	3.1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1 on page 142
ERROPFGF	32	3.2.7 ERROPFGF, Error Pseudo Fault Generation Feature Register on page 200	3.1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1 on page 144

### 3.2.2 ERROCTL, Error Record Control Register

The ERROCTL contains enable bits for the node that writes to this record:

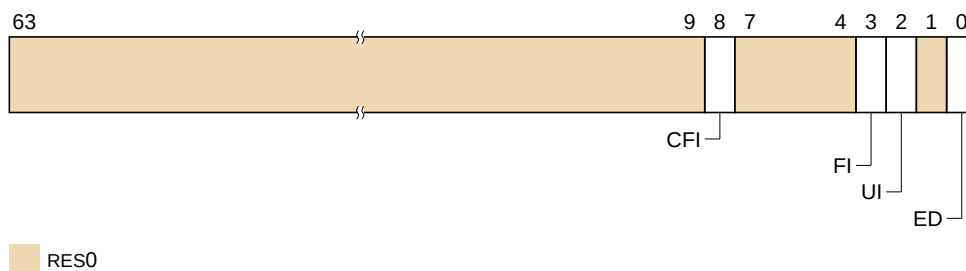
- Enabling error detection and correction.
- Enabling an error recovery interrupt.
- Enabling a fault handling interrupt.
- Enabling error recovery reporting as a read or write error response.

#### Bit field descriptions

ERROCTL is a 64-bit register and is part of the *Reliability, Availability, Serviceability* (RAS) registers functional group.

ERROCTL resets to ED is 0x0. CFI [8], FI [3], and UI [2] are **UNKNOWN**. The rest of the register is RES0.

**Figure 3-61: ERROCTL bit assignments**



#### RES0, [63:9]

RES0 Reserved.

#### CFI, [8]

Fault handling interrupt for corrected errors enable.

The fault handling interrupt is generated when one of the standard CE counters on ERRORMISCO overflows and the overflow bit is set. The possible values are:

- |   |  |
|---|--|
| 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 status is overwritten because the error record already records a higher priority error.



This applies to both reads and writes.



**RES0, [7:4]**

<b>RES0</b>	Reserved.
-------------	-----------

**FI, [3]**

Fault handling interrupt enable.

The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. The possible values are:

0	Fault handling interrupt disabled.
1	Fault handling interrupt enabled.

**UI, [2]**

Uncorrected error recovery interrupt enable. When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred. The possible values are:

0	Error recovery interrupt disabled.
1	Error recovery interrupt enabled.



Applies to both reads and writes.

---

**RES0, [1]**

<b>RES0</b>	Reserved.
-------------	-----------

**ED, [0]**

Error Detection and correction enable. In Lock-mode, this bit is **RES0**. In Split-mode, the possible values are:

0	Error detection and correction disabled.
1	Error detection and correction enabled.

**Configurations**

When ERRSELR\_EL1.SEL==0, this register is accessible from [3.1.37 ERXCTLR\\_EL1, Selected Error Record Control Register, EL1](#) on page 140.

### 3.2.3 ERROFR, Error Record Feature Register

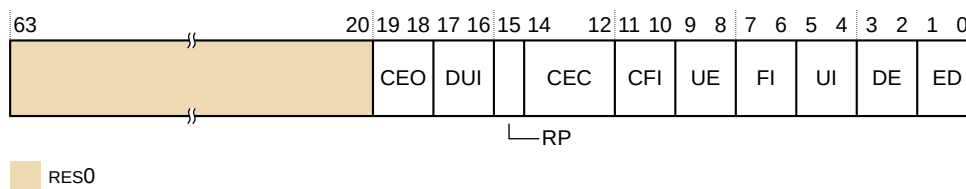
The ERROFR defines which of the common architecturally defined features are implemented and, of the implemented features, which are software programmable.

#### Bit field descriptions

ERROFR is a 64-bit register, and is part of the *Reliability, Availability, Serviceability* (RAS) registers functional group.

The register is read-only.

**Figure 3-62: ERROFR bit assignments**



#### RES0, [63:20]

RES0

Reserved.

#### CEO, [19:18]

Corrected Error Overwrite. The value is:

00 Counts CE if a counter is implemented and keeps the previous error status. If the counter overflows, ERROSTATUS.OF is set to 1.

#### DUI, [17:16]

Error recovery interrupt for deferred errors. The value is:

00 The core does not support this feature.

#### RP, [15]

Repeat counter. The value is:

1 A first repeat counter and a second other counter are implemented. The repeat counter is the same size as the primary error counter.

#### CEC, [14:12]

Corrected Error Counter. The value is:

010 The node implements an 8-bit standard CE counter in ERRORMISCO[39:32].

**CFI, [11:10]**

Fault handling interrupt for corrected errors. The value is:

- |    |   |
|----|---|
| 10 | The node implements a control for enabling fault handling interrupts on corrected errors. |
|----|---|

**UE, [9:8]**

In-band uncorrected error reporting. The value is:

- |    |   |
|----|---|
| 01 | The node implements in-band uncorrected error reporting, that is External aborts. |
|----|---|

**FI, [7:6]**

Fault handling interrupt. The value is:

- |    |  |
|----|--|
| 10 | The node implements a fault handling interrupt and implements controls for enabling and disabling. |
|----|--|

**UI, [5:4]**

Error recovery interrupt for uncorrected errors. The value is:

- |    |   |
|----|---|
| 01 | The node always enables uncorrected error recovery interrupt. |
|----|---|

**DE, [3:2]**

Defers errors. The value is:

- |           |   |
|-----------|---|
| <b>01</b> | Defers errors is always enabled for the node. |
|-----------|---|

**ED, [1:0]**

Error detection and correction.

In Lock-mode, the value is:

- |    |  |
|----|--|
| 01 | Error detection and correction is always enabled for the node. |
|----|--|

In Split-mode, the value is:

- |    |  |
|----|--|
| 10 | The node implements controls for enabling or disabling error detection and correction. |
|----|--|

**Configurations**

In Lock-mode, ERROFR resets to 0x000000000000A995

In Split-mode, ERROFR resets to 0x000000000000A996

When ERRSELR\_EL1.SEL==0, ERROFR is accessible from [3.1.38 ERXFR\\_EL1, Selected Error Record Feature Register, EL1](#) on page 140.

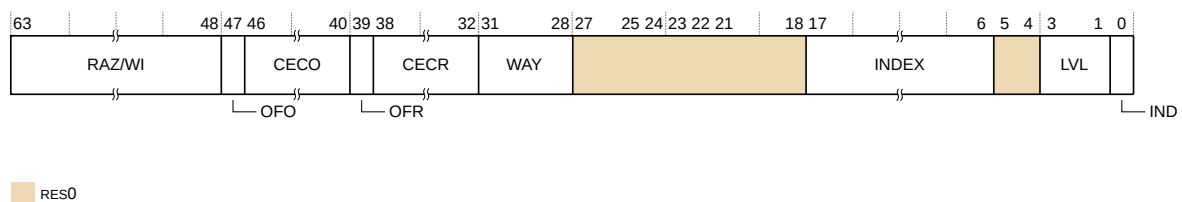
### 3.2.4 ERRORMISCO, Error Record Miscellaneous Register 0

The ERRORMISCO is an error syndrome register. It contains corrected error counters, information to identify where the error was detected, and other state information not present in the corresponding status and address error record registers. In Lock-mode or when CPUECTLR.RAS\_RAZ is set, this register is RAZ.

#### Bit field descriptions

ERRORMISCO is a 64-bit register, and is part of the *Reliability, Availability, Serviceability* (RAS) registers functional group.

**Figure 3-63: ERRORMISCO bit assignments**



#### [63:48]

**RES0** Reserved.

#### OFO, [47]

Sticky overflow bit, other. The possible values of this bit are:

- |   |                                   |
|---|-----------------------------------|
| 0 | Other counter has not overflowed. |
| 1 | Other counter has overflowed.     |

The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and either overflow bit is set to 1.

#### CECO, [46:40]

Corrected error count, other. Incremented for each Corrected error that does not match the recorded syndrome.

This field resets to an **IMPLEMENTATION DEFINED** value which might be **UNKNOWN** on a Cold reset. If the reset value is **UNKNOWN**, then the value of this field remains **UNKNOWN** until software initializes it.

#### OFR, [39]

Sticky overflow bit, repeat. The possible values of this bit are:

- |   |                                    |
|---|------------------------------------|
| 0 | Repeat counter has not overflowed. |
| 1 | Repeat counter has overflowed.     |

The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and either overflow bit is set to 1.

**CECR, [38:32]**

Corrected error count, repeat. Incremented for the first recorded error, which also records other syndrome, and then again for each Corrected error that matches the recorded syndrome.

This field resets to an **IMPLEMENTATION DEFINED** which might be **UNKNOWN** on a Cold reset. If the reset value is **UNKNOWN**, then the value of this field remains **UNKNOWN** until software initializes it.

**WAY, [31:28]**

Indicates the way that contained the error.

- For the L3 cache, all four bits are used
- For the L2 cache, L1 data cache, L1 instruction cache, and L2 TLB, only bits [31:30] are used

**[27:18]**

**RES0** Reserved.

**INDX, [17:6]**

Indicates the index that contained the error.

Upper bits of the index are unused depending on the cache size.

**[5:4]**

**RES0** Reserved.

**LVL, [3:1]**

Indicates the level that contained the error. The possible values are:

000	Level 1.
001	Level 2.

**IND, [0]**

Indicates the type of cache that contained the error. The possible values are:

0	L1 data cache, unified L2 cache, or TLB.
1	L1 instruction cache.

**Configurations**

ERRMISCO resets to [63:32] is 0x00000000, [31:0] is **UNKNOWN**.

When ERRSELR\_EL1.SEL==0, this register is accessible from [3.1.39 ERXMISCO\\_EL1, Selected Error Record Miscellaneous Register 0, EL1](#) on page 141.

The following observations should be made about this register:

- If two or more memory errors occur in the same cycle, only one error is reported but the other error count will be incremented.

- If two or more first memory error events from different RAMs occur in the same cycle, one of the errors is selected arbitrarily.
- If a new error arrives while the ERXSTATUS\_EL1.V bit is set, the way, index, and level information is not updated, but the other error field or the repeat error field is updated.
- If two or more memory errors from different RAMs occur in the same cycle when the ERXSTATUS\_EL1.V bit is set, and they do not match the level, way and index information in this register, the Other error count field is only incremented once.
- This register is not reset on a Warm reset.

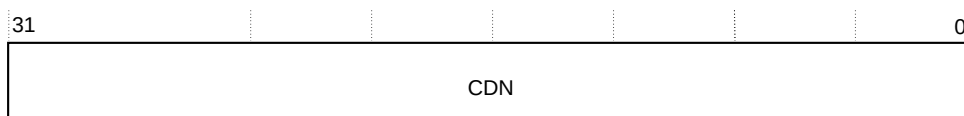
### 3.2.5 ERR0PFGCDN, Error Pseudo Fault Generation Count Down Register

ERR0PFGCDN is the Cortex®-A65AE node register that generates one of the errors that are enabled in the corresponding ERR0PFGCTL register.

## Bit field descriptions

ERRORFGCDN is a 32-bit register and is RW.

### Figure 3-64: ERR0PFGCDN bit assignments



CDN, [31:0]

Count Down value. The reset value of the Error Generation Counter is used for the countdown.

## Configurations

There are no configuration options.

ERRORFGCDN resets to **UNKNOWN**.

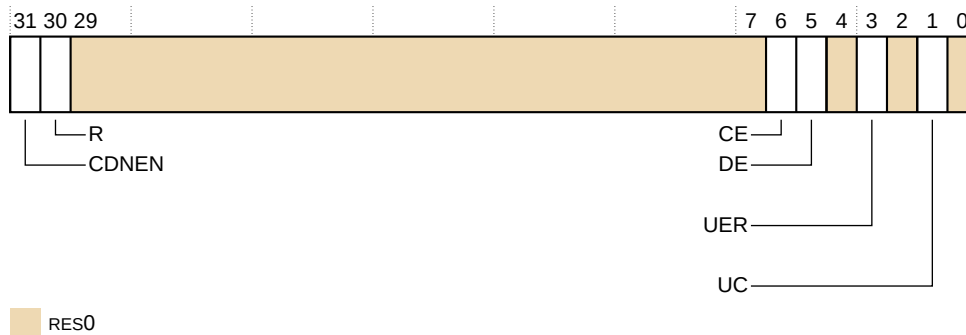
When ERRSEL\_EL1.SEL==0, ERRPGCDN is accessible from [3.1.40 ERXPGCDN\\_EL1](#), [Selected Error Pseudo Fault Generation Count Down Register, EL1](#) on page 141.

### 3.2.6 ERR0PFGCTL, Error Pseudo Fault Generation Control Register

The `ERR0PFGCTL` is the Cortex®-A65AE node register that enables controlled fault generation.

## Bit field descriptions

ERRPFGCTL is a 32-bit register and is RW.

**Figure 3-65: ERR0PFGCTL bit assignments****CDNEN, [31]**

Count down enable. This bit controls transfers from the value that is held in the ERR0PFGCDN into the Error Generation Counter and enables this counter to start counting down. The possible values are:

- |   |   |
|---|---|
| 0 | The Error Generation Counter is disabled. This is the reset value.  |
| 1 | The value that is held in the ERR0PFGCDN register is transferred into the Error Generation Counter. The Error Generation Counter counts down. |

**R, [30]**

Restartable bit. When it reaches 0, the Error Generation Counter restarts from the ERR0PFGCDN value or stops. The possible values are:

- |   |   |
|---|---|
| 0 | When it reaches 0, the counter stops. This is the reset value.  |
| 1 | When it reaches 0, the counter reloads the value that is stored in ERR0PFGCDN and starts counting down again. |

**[29:7]**

Reserved, **UNK/SBZP**.

**CE, [6]**

Corrected error generation enable. The possible values are:

- |   |  |
|---|--|
| 0 | No corrected error is generated. This is the reset value.                            |
| 1 | A corrected error might be generated when the Error Generation Counter is triggered. |

**DE, [5]**

Deferred Error generation enable. The possible values are:

- |   |  |
|---|--|
| 0 | No deferred error is generated. This is the reset value. |
|---|--|

- |   |   |
|---|---|
| 1 | A deferred error might be generated when the Error Generation Counter is triggered. |
|---|---|

**RES0, [4]**

- |      |           |
|------|-----------|
| RES0 | Reserved. |
|------|-----------|

**UER, [3]**

Signaled or Recoverable Error generation. The value is:

- |   |   |
|---|---|
| 0 | No signaled or recoverable error is generated. This is the reset value. |
| 1 | This feature is controllable.   |

**RES0, [2]**

- |      |           |
|------|-----------|
| RES0 | Reserved. |
|------|-----------|

**UC, [1]**

Uncontainable error generation enable. The possible values are:

- |   |   |
|---|---|
| 0 | No uncontainable error is generated. This is the reset value.                             |
| 1 | An uncontainable error might be generated when the Error Generation Counter is triggered. |

**[0]**

- |  |                             |
|--|-----------------------------|
|  | Reserved, <b>UNK/SBZP</b> . |
|--|-----------------------------|

**Configurations**

There are no configuration notes.

ERR0PFGCTL resets to **UNKNOWN**.

When ERRSELR\_EL1.SEL==0, ERR0PFGCTL is accessible from [3.1.41 ERXPFGCTL\\_EL1, Selected Error Pseudo Fault Generation Control Register, EL1](#) on page 142.

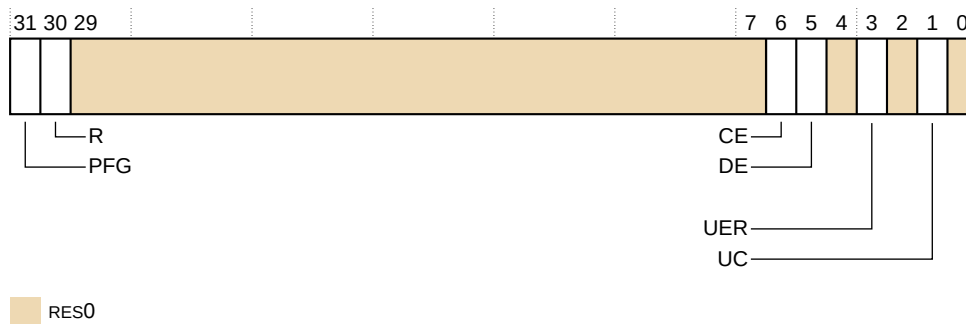
**3.2.7 ERR0PFGF, Error Pseudo Fault Generation Feature Register**

The ERR0PFGF is the Cortex®-A65AE node register that defines which fault generation features are implemented.

**Bit field descriptions**

ERR0PFGF is a 32-bit register and is RO.



**Figure 3-66: ERR0PFGF bit assignments****PFG, [31]**

Pseudo Fault Generation. The value is:

- 1            The node implements a fault injection mechanism.

**R, [30]**

Restartable bit. When it reaches zero, the Error Generation Counter restarts from the ERR0PFGCDN value or stops. The value is:

- 1            This feature is controllable.

**[29:7]**

Reserved, **UNK/SBZP**.

**CE, [6]**

Corrected Error generation. The value is:

- 1            This feature is controllable.

**DE, [5]**

Deferred Error generation. The value is:

- 1            This feature is controllable.

**RES0, [4]**

- RES0**       Reserved.

**UER, [3]**

Signaled or Recoverable Error generation. The value is:

- 1            This feature is controllable.

**RES0, [2]****RES0** Reserved.**UC, [1]**

Uncontainable Error generation. The value is:

1 This feature is controllable.

**[0]**Reserved, **UNK/SBZP**.**Configurations**

There are no configuration notes.

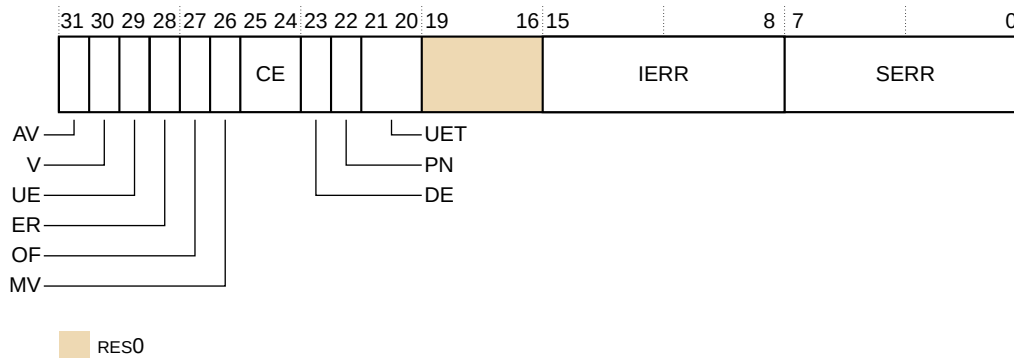
ERROPFGF resets to **UNKNOWN**.When ERRSELR\_EL1.SEL==0, ERROPFGF is accessible from [3.1.42 ERXPFGF\\_EL1, Selected Pseudo Fault Generation Feature Register, EL1](#) on page 144.**3.2.8 ERROSTATUS, Error Record Primary Status Register**

The ERROSTATUS contains information about the error record:

- Whether any error has been detected.
- Whether any detected error was not corrected and returned to a master.
- Whether any detected error was not corrected and deferred.
- Whether a second error of the same type was detected before software handled the first error.
- Whether any error has been reported.
- Whether the other error record registers contain valid information.
- In Lock-mode or when CPUECTLR.RAS\_RAZ is set, this register is RAZ.

**Bit field descriptions**

ERROSTATUS is a 32-bit register.

**Figure 3-67: ERR0STATUS bit assignments****AV, [31]**

Address Valid. The value is:

0            ERROADDR is not valid.

**V, [30]**

Status Register valid. The possible values are:

0            ERR0STATUS is not valid. This is the reset value.  
 1            ERR0STATUS is valid. At least one error has been recorded.

**UE, [29]**

Uncorrected error. The possible values are:

0            No error that could not be corrected or deferred has been detected.  
              This is the reset value.  
 1            At least one error that could not be corrected or deferred has been  
              detected. If error recovery interrupts are enabled, then the interrupt  
              signal is asserted until this bit is cleared.

**ER, [28]**

Error reported. The possible values are:

0            No external abort has been reported. This is the reset value.  
 1            The node has reported an external abort to the master that is in access  
              or making a transaction.

**OF, [27]**

Overflow. The possible values are:

0            • If UE == 1, then no error status for an Uncorrected error has been  
              discarded.

- If UE == 0 and DE == 1, then no error status for a Deferred error has been discarded.
- If UE == 0, DE == 0, and CE != 0x00, then:
  - If a Corrected error counter is implemented, it has not overflowed.
  - If no Corrected error counter is implemented, no error status for a Corrected error has been discarded.

1 This is the reset value.  
More than one error has occurred and so details of the other error have been discarded.

**MV, [26]**

Miscellaneous Registers Valid. The possible values are:

0 ERRORMISCO and ERRORMISC1 are not valid. This is the reset value.  
1 This bit indicates that ERRORMISCO contains additional information about any error recorded by this record.

**CE, [25:24]**

Corrected error. The possible values are:

0x0 No corrected errors recorded. This is the reset value.  
0x2 At least one corrected error recorded.

**DE, [23]**

Deferred error. The possible values are:

0 No errors were deferred. This is the reset value.  
1 At least one error was not corrected and deferred by poisoning.

**PN, [22]**

Poison. The value is:

0 No uncorrected errors were detected. This is the reset value.  
1 At least one uncorrected error detected by reading a poisoned line.

**UI, CT, [21:20]**

Uninfected and Containable Error Type. The value is:

0x0 Uncontainable. This is the reset value.

**RES0, [19:16]**

RES0 Reserved.

**IERR, [15:8]**

**IMPLEMENTATION DEFINED** error code. The possible values are:

0x00	No error, or error on other RAMs.
0x01	Error on L1 dirty RAM.
0x02	Error on L3 snoop filter RAM.

**SERR, [7:0]**

Primary error code. The possible values are:

0x00	No error.
0x02	ECC error from internal data buffer.
0x06	ECC error on cache data RAM.
0x07	ECC error on cache tag or dirty RAM.
0x08	Parity error on TLB data RAM.
0x09	Parity error on TLB tag RAM.
0x12	Bus error.
0x15	Deferred error from slave not supported at the consumer. For example, poisoned data received from a slave by a master that cannot defer the error further.

**Configurations**

There are no configuration notes.

ERROSTATUS resets to 0x0000000000000000.

When ERRSELR\_EL1.SEL==0, ERROSTATUS is accessible from [3.1.43 ERXSTATUS\\_EL1, Selected Error Record Primary Status Register, EL1](#) on page 145.

The following observations should be made about this register:

- When this record is written, the corresponding **nFAULTIRQ** pin will be asserted if the error is uncorrectable or if the correctable error counter has overflowed (if enabled in the ERXCTLR). To deassert the pin, software must set the Valid field to zero.
- When the UE field is set for an error on an L3 RAM, the **nERRIRQ[0]** pin is asserted (if enabled in the ERXCTLR). To deassert the pin, software must set the UE field to zero.
- When the UE field is set for an error on a L1 or L2 RAM, the **nERRIRQ[n+1]** pin is asserted for core n (if enabled in the ERXCTLR). To deassert the pin, software must set the UE field to zero.
- This register is not reset on a Warm reset.

## 3.3 GIC registers

This chapter describes the GIC registers.

### 3.3.1 CPU interface registers

Each CPU interface block provides the interface for the Cortex®-A65AE core that interfaces with a GIC distributor within the system.

The Cortex®-A65AE core only supports System register access to the GIC CPU interface registers. The following table lists the three types of GIC CPU interface System registers supported in the Cortex®-A65AE core.

**Table 3-40: GIC CPU interface System register types supported in the Cortex®-A65AE core.**

Register prefix	Register type
ICC	Physical GIC CPU interface System registers.
ICV	Virtual GIC CPU interface System registers.
ICH	Virtual interface control System registers.

Access to virtual GIC CPU interface System registers is only possible at Non-secure EL1.

Access to ICC registers or the equivalent ICV registers is determined by HCR\_EL2. See [3.1.48 HCR\\_EL2, Hypervisor Configuration Register, EL2](#) on page 148.

For more information on the CPU interface, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

### 3.3.2 AArch64 physical GIC CPU interface System register summary

The following table lists the AArch64 physical GIC CPU interface System registers that have **IMPLEMENTATION DEFINED** bits.

See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information and a complete list of AArch64 physical GIC CPU interface System registers.

**Table 3-41: AArch64 physical GIC CPU interface System register summary**

Name	Op0	Op1	CRn	CRm	Op2	Type	Description
ICC_AP0R0_EL1	3	0	12	8	4	RW	<a href="#">3.3.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Register 0, EL1</a> on page 207
ICC_AP1R0_EL1	3	0	12	9	0	RW	<a href="#">3.3.4 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Register 0 EL1</a> on page 207
ICC_BPR0_EL1	3	0	12	8	3	RW	<a href="#">3.3.5 ICC_BPR0_EL1, Interrupt Controller Binary Point Register 0, EL1</a> on page 208
ICC_BPR1_EL1	3	0	12	12	3	RW	<a href="#">3.3.6 ICC_BPR1_EL1, Interrupt Controller Binary Point Register 1, EL1</a> on page 208
ICC_CTLR_EL1	3	0	12	12	4	RW	<a href="#">3.3.7 ICC_CTLR_EL1, Interrupt Controller Control Register, EL1</a> on page 209
ICC_CTLR_EL3	3	6	12	12	4	RW	<a href="#">3.3.8 ICC_CTLR_EL3, Interrupt Controller Control Register, EL3</a> on page 211
ICC_SRE_EL1	3	0	12	12	5	RW	<a href="#">3.3.9 ICC_SRE_EL1, Interrupt Controller System Register Enable Register, EL1</a> on page 213

Name	Op0	Op1	CRn	CRm	Op2	Type	Description
ICC_SRE_EL2	3	4	12	9	5	RW	<a href="#">3.3.10 ICC_SRE_EL2, Interrupt Controller System Register Enable register, EL2 on page 215</a>
ICC_SRE_EL3	3	6	12	12	5	RW	<a href="#">3.3.11 ICC_SRE_EL3, Interrupt Controller System Register Enable register, EL3 on page 216</a>

### 3.3.3 ICC\_AP0R0\_EL1, Interrupt Controller Active Priorities Group 0 Register 0, EL1

The ICC\_AP0R0\_EL1 provides information about Group 0 active priorities.

#### Bit descriptions

This register is a 32-bit register and is part of:

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

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

0x00000000 No interrupt active. This is the reset value.

0x00000001 Interrupt active for priority 0x0.

0x00000002 Interrupt active for priority 0x8.

...

0x80000000 Interrupt active for priority 0xF8.

### 3.3.4 ICC\_AP1R0\_EL1, Interrupt Controller Active Priorities Group 1 Register 0 EL1

The ICC\_AP1R0\_EL1 provides information about Group 1 active priorities.

#### Bit descriptions

This register is a 32-bit register and is part of:

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

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

0x00000000 No interrupt active. This is the reset value.

0x00000001 Interrupt active for priority 0x0.

0x00000002 Interrupt active for priority 0x8.

...

0x80000000 Interrupt active for priority 0xF8.

### 3.3.5 ICC\_BPR0\_EL1, Interrupt Controller Binary Point Register 0, EL1

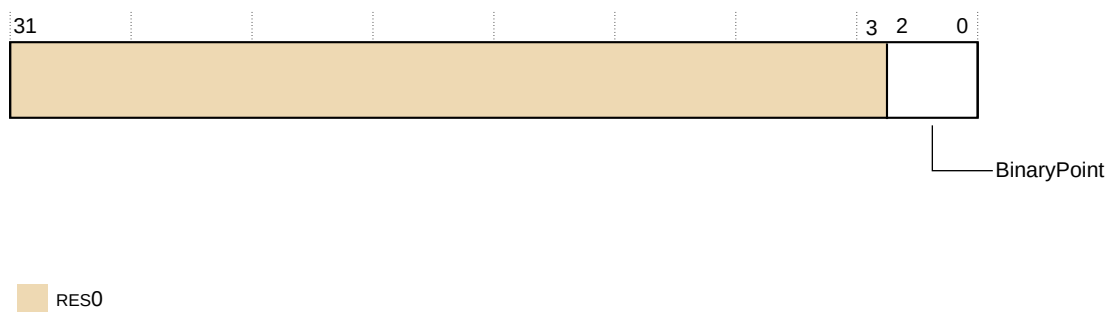
ICC\_BPR0\_EL1 defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 0 interrupt preemption.

#### Bit field descriptions

ICC\_BPR0\_EL1 is a 32-bit register and is part of:

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

**Figure 3-68: ICC\_BPR0\_EL1 bit assignments**



#### RES0, [31:3]

**RES0** Reserved.

#### BinaryPoint, [2:0]

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. The minimum value that is implemented is:

**0x2**

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.



### 3.3.6 ICC\_BPR1\_EL1, Interrupt Controller Binary Point Register 1, EL1

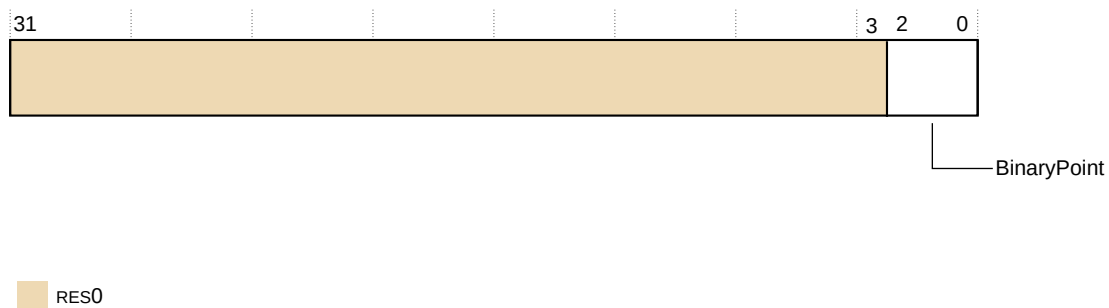
ICC\_BPR1\_EL1 defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption.

#### Bit field descriptions

ICC\_BPR1\_EL1 is a 32-bit register and is part of:

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

**Figure 3-69: ICC\_BPR1\_EL1 bit assignments**



#### RES0, [31:3]

**RES0** Reserved.

#### BinaryPoint, [2:0]

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field.

The minimum value that is implemented of ICC\_BPR1\_EL1 Secure register is 0x2.

The minimum value that is implemented of ICC\_BPR1\_EL1 Non-secure register is 0x3.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

### 3.3.7 ICC\_CTLR\_EL1, Interrupt Controller Control Register, EL1

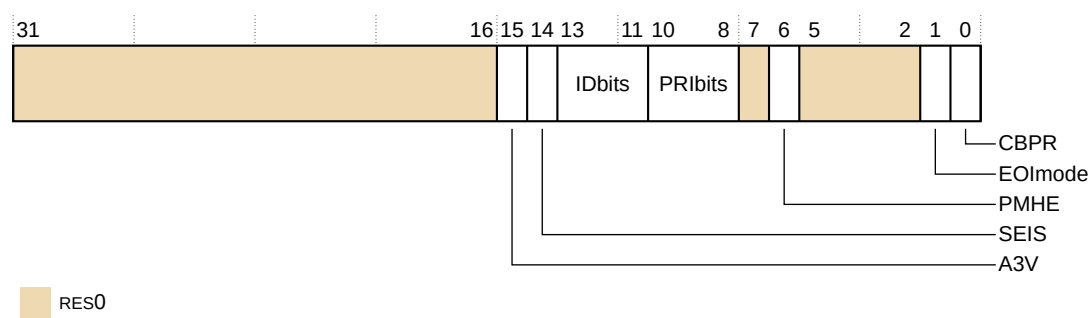
ICC\_CTLR\_EL1 controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

#### Bit field descriptions

ICC\_CTLR\_EL1 is a 32-bit register and is part of:

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

Figure 3-70: ICC\_CTLR\_EL1 bit assignments



RES0, [31:16]

RES0      Reserved.

A3V, [15]

Affinity 3 Valid. The value is:

1      The CPU interface logic supports nonzero values of Affinity 3 in SGI generation System registers.

SEIS, [14]

SEI Support. The value is:

0      The CPU interface logic does not support local generation of SEIs.

IDbits, [13:11]

Identifier bits. The value is:

0      The number of physical interrupt identifier bits supported is 16 bits.

This field is an alias of ICC\_CTLR\_EL3.IDbits.

PRIbits, [10:8]

Priority bits. The value is:

0x4      The core supports 32 levels of physical priority with 5 priority bits.

RES0, [7]

RES0      Reserved.

**PMHE, [6]**

0	Disables use of ICC_PMR as a hint for interrupt distribution.
1	Enables use of ICC_PMR as a hint for interrupt distribution.

**RES0, [5:2]**

<b>RES0</b>	Reserved.
-------------	-----------

**EOImode, [1]**

End of interrupt mode for the current Security state. The possible values are:

0	ICC_EOIR0 and ICC_EOIR1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR are <b>UNPREDICTABLE</b> .
1	ICC_EOIR0 and ICC_EOIR1 provide priority drop functionality only. ICC_DIR provides interrupt deactivation functionality.

**CBPR, [0]**

Common Binary Point Register. Control whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupt. The possible values are:

0	ICC_BPR0 determines the preemption group for Group 0 interrupts.
1	ICC_BPR1 determines the preemption group for Group 1 interrupts. ICC_BPR0 determines the preemption group for Group 0 and Group 1 interrupts.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

**3.3.8 ICC\_CTLR\_EL3, Interrupt Controller Control Register, EL3**

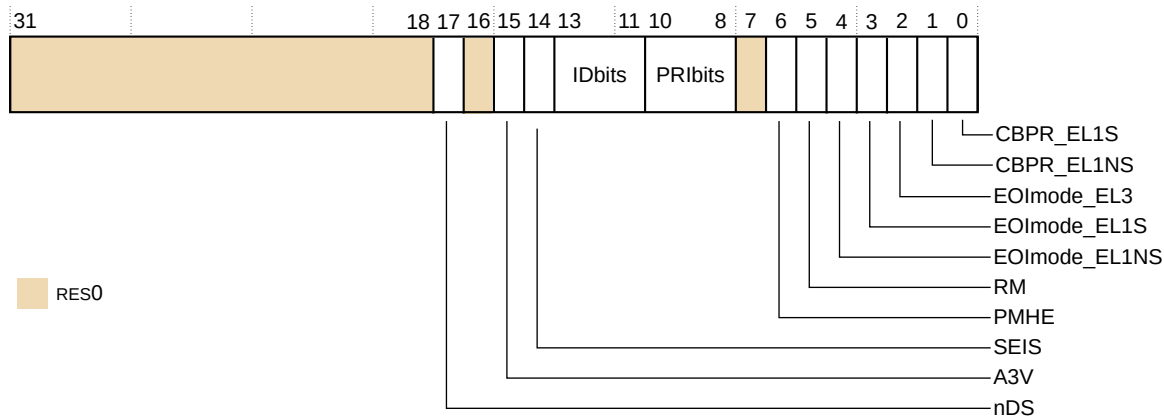
ICC\_CTLR\_EL3 controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

**Bit field descriptions**

ICC\_CTLR\_EL3 is a 32-bit register and is part of:

- The GIC System registers functional group.
- The Security registers functional group.
- The GIC control registers functional group.

Figure 3-71: ICC\_CTLR\_EL3 bit assignments



RES0, [31:18]

RES0 Reserved.

nDS, [17]

Disable Security not supported. Read-only and writes are **IGNORED**. The value is:

1 The CPU interface logic does not support disabling of security, and requires that security is not disabled.

RES0, [16]

RES0 Reserved.

A3V, [15]

Affinity 3 Valid. This bit is RAO/WI.

SEIS, [14]

SEI Support. The value is:

0 The CPU interface logic does not support generation of SEIs.

IDbits, [13:11]

Identifier bits. The value is:

0x0 The number of physical interrupt identifier bits supported is 16 bits.

This field is an alias of ICC\_CTLR\_EL3.IDbits.

PRIbits, [10:8]

Priority bits. The value is:

0x4      The core supports 32 levels of physical priority with 5 priority bits.

## RES0, [7]

Reserved, **RES0**.

## PMHE, [6]

Priority Mask Hint Enable. The possible values are:

- |   |   |
|---|---|
| 0 | Disables use of ICC_PMR as a hint for interrupt distribution. |
| 1 | Enables use of ICC_PMR as a hint for interrupt distribution.  |

## RM, [5]

Routing Modifier. This bit is RAZ/WI.

## EOImode\_EL1NS, [4]

EOI mode for interrupts handled at Non-secure EL1 and EL2.

Controls whether a write to an End of Interrupt register also deactivates the interrupt.

## EOImode\_EL1S, [3]

EOI mode for interrupts handled at Secure EL1.

Controls whether a write to an End of Interrupt register also deactivates the interrupt.

## EOImode\_EL3, [2]

EOI mode for interrupts handled at EL3.

Controls whether a write to an End of Interrupt register also deactivates the interrupt.

## CBPR\_EL1NS, [1]

Common Binary Point Register, EL1 Non-secure.

Control whether the same register is used for interrupt preemption of both Group 0 and Group 1 Non-secure interrupts at EL1 and EL2.

## CBPR\_EL1S, [0]

Common Binary Point Register, EL1 Secure.

Control whether the same register is used for interrupt preemption of both Group 0 and Group 1 Secure interrupt at EL1.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

### 3.3.9 ICC\_SRE\_EL1, Interrupt Controller System Register Enable Register, EL1

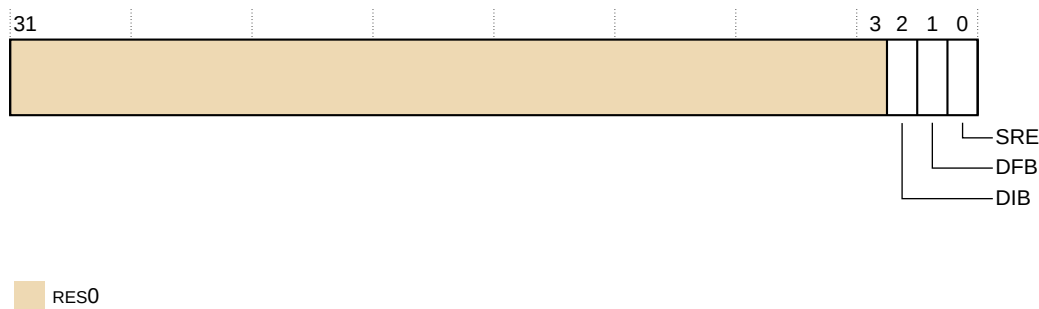
ICC\_SRE\_EL1 controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL0 and EL1.

#### Bit field descriptions

ICC\_SRE\_EL1 is a 32-bit register and is part of:

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

**Figure 3-72: ICC\_SRE\_EL1 bit assignments**



#### RES0, [31:3]

**RES0**      Reserved.

#### DIB, [2]

Disable IRQ bypass. The possible values are:

0x0	IRQ bypass enabled.
0x1	IRQ bypass disabled.

This bit is an alias of ICC\_SRE\_EL3.DIB

#### DFB, [1]

Disable FIQ bypass. The possible values are:

0x0	FIQ bypass enabled.
0x1	FIQ bypass disabled.

This bit is an alias of ICC\_SRE\_EL3.DFB

#### SRE, [0]

System Register Enable. The value is:

0x1	The System register interface for the current Security state is enabled.
-----	--

This bit is RAO/WI. The core only supports a System register interface to the GIC CPU interface.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

### 3.3.10 ICC\_SRE\_EL2, Interrupt Controller System Register Enable register, EL2

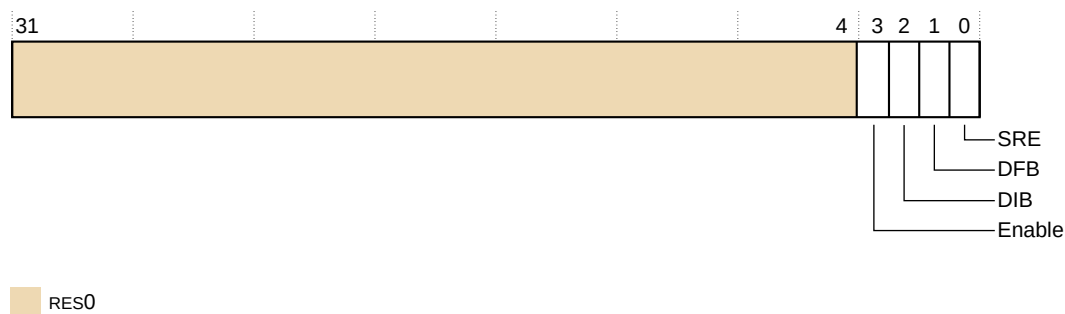
ICC\_SRE\_EL2 controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL2.

#### Bit field descriptions

ICC\_SRE\_EL2 is a 32-bit register and is part of:

- The GIC System registers functional group.
- The Virtualization registers functional group.
- The GIC control registers functional group.

**Figure 3-73: ICC\_SRE\_EL2 bit assignments**



#### RES0, [31:4]

**RES0** Reserved.

#### Enable, [3]

Enables lower Exception level access to ICC\_SRE\_EL1. The value is:

**0x1** Non-secure EL1 accesses to ICC\_SRE\_EL1 do not trap to EL2.

This bit is RAO/WI.

#### DIB, [2]

Disable IRQ bypass. The possible values are:

**0x0** IRQ bypass enabled.

0x1          IRQ bypass disabled.

This bit is an alias of ICC\_SRE\_EL3.DIB

#### **DFB, [1]**

Disable FIQ bypass. The possible values are:

0x0          FIQ bypass enabled.  
0x1          FIQ bypass disabled.

This bit is an alias of ICC\_SRE\_EL3.DFB

#### **SRE, [0]**

System Register Enable. The value is:

0x1          The System register interface for the current Security state is enabled.

This bit is RAO/WI. The core only supports a System register interface to the GIC CPU interface.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

### **3.3.11 ICC\_SRE\_EL3, Interrupt Controller System Register Enable register, EL3**

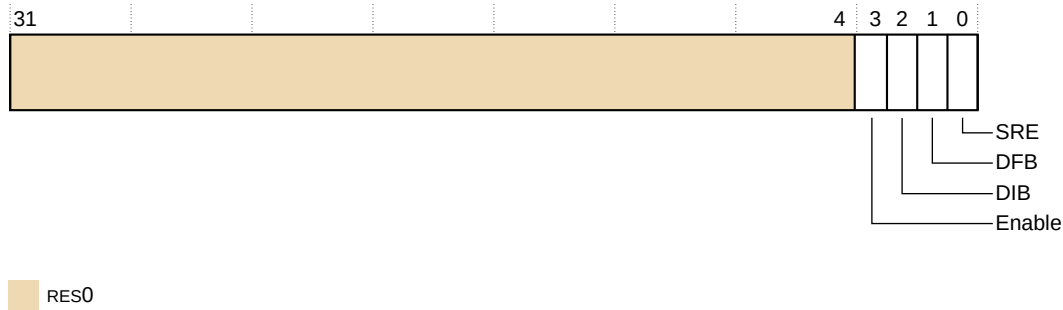
ICC\_SRE\_EL3 controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL3.

#### **Bit field descriptions**

ICC\_SRE\_EL3 is a 32-bit register and is part of:

- The GIC System registers functional group.
- The Security registers functional group.
- The GIC control registers functional group.



**Figure 3-74: ICC\_SRE\_EL3 bit assignments****RES0, [31:4]**

**RES0** Reserved.

**Enable, [3]**

Enables lower Exception level access to ICC\_SRE\_EL1 and ICC\_SRE\_EL2. The value is:

- |   |   |
|---|---|
| 1 | <ul style="list-style-type: none"> <li>Secure EL1 accesses to Secure ICC_SRE_EL1 do not trap to EL3.</li> <li>EL2 accesses to Non-secure ICC_SRE_EL1 and ICC_SRE_EL2 do not trap to EL3.</li> <li>Non-secure EL1 accesses to ICC_SRE_EL1 do not trap to EL3.</li> </ul> |
|---|---|

This bit is RAO/WI.

**DIB, [2]**

Disable IRQ bypass. The possible values are:

- |   |                      |
|---|----------------------|
| 0 | IRQ bypass enabled.  |
| 1 | IRQ bypass disabled. |

**DFB, [1]**

Disable FIQ bypass. The possible values are:

- |   |                      |
|---|----------------------|
| 0 | FIQ bypass enabled.  |
| 1 | FIQ bypass disabled. |

**SRE, [0]**

System Register Enable. The value is:

- |   |  |
|---|--|
| 1 | The System register interface for the current Security state is enabled. |
|---|--|

This bit is RAO/WI. The core only supports a System register interface to the GIC CPU interface.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

### 3.3.12 AArch64 virtual GIC CPU interface register summary

The following table describes the AArch64 virtual GIC CPU interface System registers that have **IMPLEMENTATION DEFINED** bits.

See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information and a complete list of AArch64 virtual GIC CPU interface System registers.

**Table 3-42: AArch64 virtual GIC CPU interface register summary**

Name	Op0	Op1	CRn	CRm	Op2	Type	Description
ICV_AP0R0_EL1	3	0	12	8	4	RW	<a href="#">3.3.13 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Register 0, EL1</a> on page 218
ICV_AP1R0_EL1	3	0	12	9	0	RW	<a href="#">3.3.14 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Register 0, EL1</a> on page 218
ICV_BPR0_EL1	3	0	12	8	3	RW	<a href="#">3.3.15 ICV_BPR0_EL1, Interrupt Controller Virtual Binary Point Register 0, EL1</a> on page 219
ICV_BPR1_EL1	3	0	12	12	3	RW	<a href="#">3.3.16 ICV_BPR1_EL1, Interrupt Controller Virtual Binary Point Register 1, EL1</a> on page 220
ICV_CTLR_EL1	3	0	12	12	4	RW	<a href="#">3.3.17 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register, EL1</a> on page 221

### 3.3.13 ICV\_AP0R0\_EL1, Interrupt Controller Virtual Active Priorities Group 0 Register 0, EL1

The ICV\_AP0R0\_EL1 register provides information about virtual Group 0 active priorities.

#### Bit descriptions

This register is a 32-bit register and is part of the virtual GIC System registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

0x00000000 No interrupt active. This is the reset value.

0x00000001 Interrupt active for priority 0x0.

0x00000002 Interrupt active for priority 0x8.

...

0x80000000 Interrupt active for priority 0xF8.

Details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

### 3.3.14 ICV\_AP1R0\_EL1, Interrupt Controller Virtual Active Priorities Group 1 Register 0, EL1

The ICV\_AP1R0\_EL1 register provides information about virtual Group 1 active priorities.

#### Bit descriptions

This register is a 32-bit register and is part of the virtual GIC System registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

0x00000000 No interrupt active. This is the reset value.

0x00000001 Interrupt active for priority 0x0.

0x00000002 Interrupt active for priority 0x8.

...

0x80000000 Interrupt active for priority 0xF8.

Details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

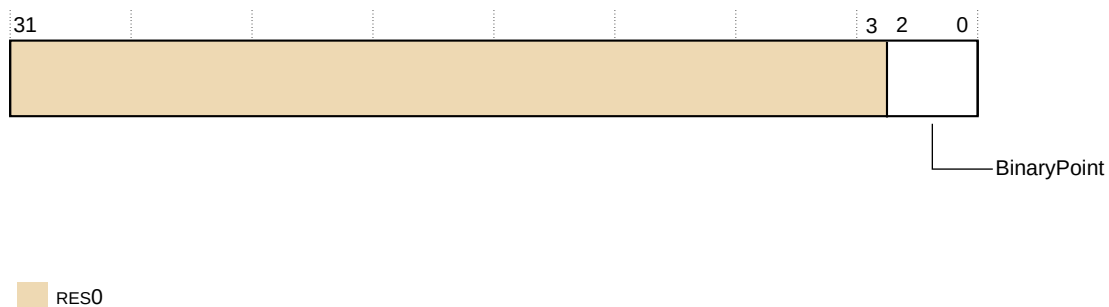
### 3.3.15 ICV\_BPR0\_EL1, Interrupt Controller Virtual Binary Point Register 0, EL1

ICV\_BPR0\_EL1 defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 0 interrupt preemption.

#### Bit field descriptions

ICC\_BPR0\_EL1 is a 32-bit register and is part of the virtual GIC System registers functional group.

**Figure 3-75: ICV\_BPR0\_EL1 bit assignments**



**RES0, [31:3]**

Reserved, **RES0**.

**BinaryPoint, [2:0]**

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. The minimum value that is implemented is:

**0x2**

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

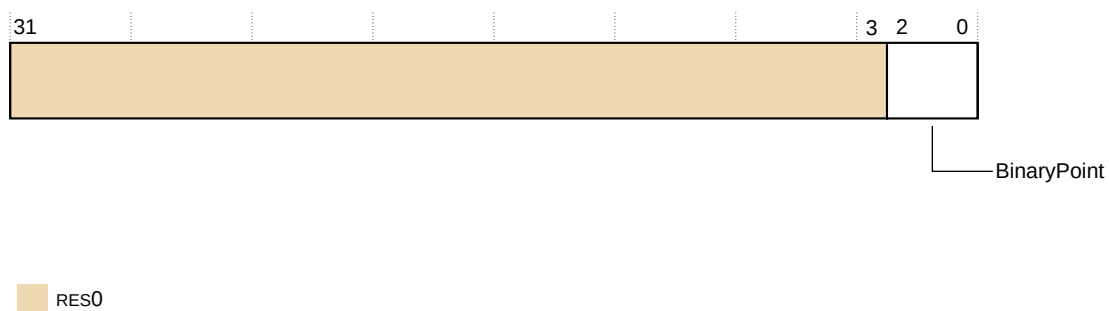
### 3.3.16 ICV\_BPR1\_EL1, Interrupt Controller Virtual Binary Point Register 1, EL1

ICV\_BPR1\_EL1 defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 1 interrupt preemption.

**Bit field descriptions**

ICV\_BPR1\_EL1 is a 32-bit register and is part of the virtual GIC System registers functional group.

**Figure 3-76: ICV\_BPR1\_EL1 bit assignments**

**RES0, [31:3]**

**RES0** Reserved.

**BinaryPoint, [2:0]**

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field.

The minimum value that is implemented of ICV\_BPR1\_EL1 Secure register is 0x2.

The minimum value that is implemented of ICV\_BPR1\_EL1 Non-secure register is 0x3.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

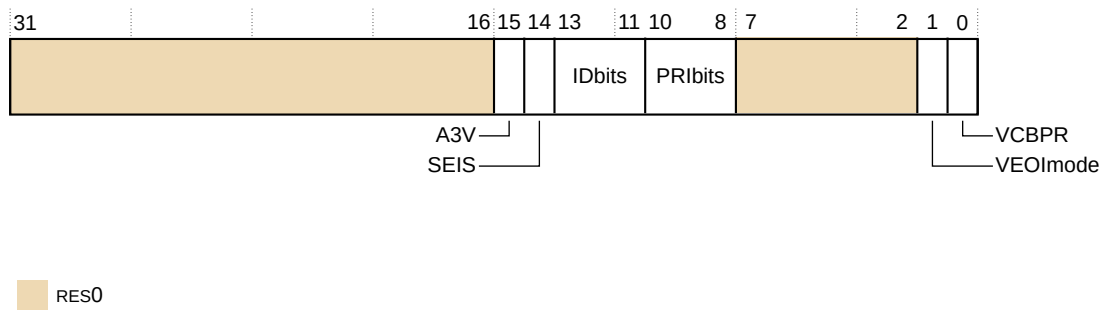
### 3.3.17 ICV\_CTLR\_EL1, Interrupt Controller Virtual Control Register, EL1

ICV\_CTLR\_EL1 controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

#### Bit field descriptions

ICV\_CTLR\_EL1 is a 32-bit register and is part of the virtual GIC System registers functional group.

**Figure 3-77: ICV\_CTLR\_EL1 bit assignments**



#### RES0, [31:16]

**RES0** Reserved.

#### A3V, [15]

Affinity 3 Valid. The value is:

**0x1** The virtual CPU interface logic supports nonzero values of Affinity 3 in SGI generation System registers.

#### SEIS, [14]

SEI Support. The value is:

**0x0** The virtual CPU interface logic does not support local generation of SEIs.

#### IDbits, [13:11]

Identifier bits. The value is:

**0x0** The number of physical interrupt identifier bits supported is 16 bits.

#### PRIbits, [10:8]

Priority bits. The value is:

**0x4** Support 32 levels of physical priority (5 priority bits).

**RES0, [7:2]**

**RES0** Reserved.

**VEOImode, [1]**

Virtual EOI mode. The possible values are:

- 0x0 ICV\_EOIR0\_EL1 and ICV\_EOIR1\_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICV\_DIR\_EL1 are **UNPREDICTABLE**.
- 0x1 ICV\_EOIR0\_EL1 and ICV\_EOIR1\_EL1 provide priority drop functionality only. ICV\_DIR provides interrupt deactivation functionality.

**VCBPR, [0]**

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts. The possible values are:

- 0 ICV\_BPRO\_EL1 determines the preemption group for virtual Group 0 interrupts only.
- 1 ICV\_BPR1\_EL1 determines the preemption group for virtual Group 1 interrupts.

Reads of ICV\_BPR1\_EL1 return ICV\_BPRO\_EL1 plus one, saturated to 111. Writes to ICV\_BPR1\_EL1 are **IGNORED**.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

**3.3.18 AArch64 virtual interface control System register summary**

The following table lists the AArch64 virtual interface control System registers that have **IMPLEMENTATION DEFINED** bits.

See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information and a complete list of AArch64 virtual interface control System registers.

**Table 3-43: AArch64 virtual interface control System register summary**

Name	Op0	Op1	CRn	CRm	Op2	Type	Description
ICH_AP0R0_EL2	3	0	12	8	4	RW	3.3.19 ICH_AP0R0_EL2, Interrupt Controller Hyp Active Priorities Group 0 Register 0, EL2 on page 223
ICH_AP1R0_EL2	3	0	19	9	0	RW	3.3.20 ICH_AP1R0_EL2, Interrupt Controller Hyp Active Priorities Group 1 Register 0, EL2 on page 223

Name	Op0	Op1	CRn	CRm	Op2	Type	Description
ICH_HCR_EL2	3	4	12	11	0	RW	<a href="#">3.3.21 ICH_HCR_EL2, Interrupt Controller Hyp Control Register, EL2</a> on page 224
ICH_VTR_EL2	3	4	12	11	1	RO	<a href="#">3.3.22 ICH_VMCR_EL2, Interrupt Controller Virtual Machine Control Register, EL2</a> on page 226
ICH_VMCR_EL2	3	4	12	11	7	RW	<a href="#">3.3.23 ICH_VTR_EL2, Interrupt Controller VGIC Type Register, EL2</a> on page 229

### 3.3.19 ICH\_AP0R0\_EL2, Interrupt Controller Hyp Active Priorities Group 0 Register 0, EL2

The ICH\_AP0R0\_EL2 provides information about Group 0 active priorities for EL2.

#### Bit field descriptions

This register is a 32-bit register and is part of:

- The GIC System registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

0x00000000 No interrupt active. This is the reset value.

0x00000001 Interrupt active for priority 0x0.

0x00000002 Interrupt active for priority 0x8.

...

0x80000000 Interrupt active for priority 0xF8.

Details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

### 3.3.20 ICH\_AP1R0\_EL2, Interrupt Controller Hyp Active Priorities Group 1 Register 0, EL2

The ICH\_AP1R0\_EL2 provides information about Group 1 active priorities for EL2.

#### Bit field descriptions

This register is a 32-bit register and is part of:

- The GIC System registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

0x00000000 No interrupt active. This is the reset value.

0x00000001 Interrupt active for priority 0x0.

0x00000002 Interrupt active for priority 0x8.

...

0x80000000 Interrupt active for priority 0xF8.

Details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

### 3.3.21 ICH\_HCR\_EL2, Interrupt Controller Hyp Control Register, EL2

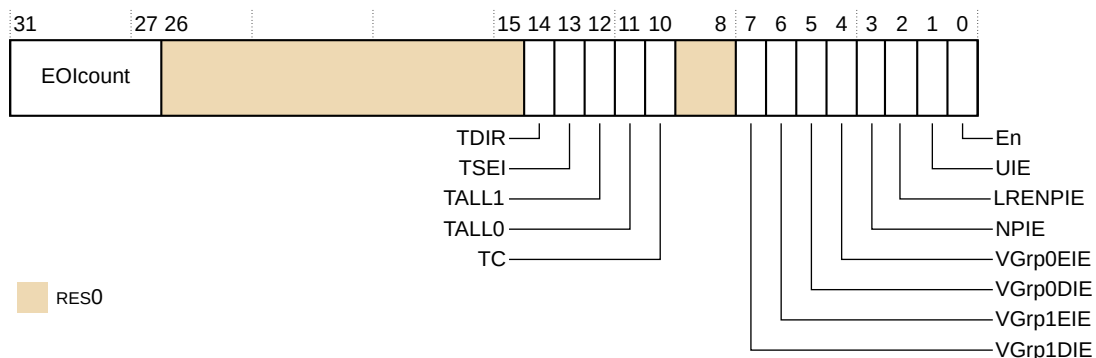
ICH\_HCR\_EL2 controls the environment for VMs.

#### Bit field descriptions

ICH\_HCR\_EL2 is a 32-bit register and is part of:

- The GIC System registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

**Figure 3-78: ICH\_HCR\_EL2 bit assignments**



#### EOIcount, [31:27]

Number of outstanding deactivates.

#### RES0, [26:15]

RES0 Reserved.

#### TDIR, [14]

Trap Non-secure EL1 writes to ICC\_DIR\_EL1 and ICV\_DIR\_EL1. The possible values are:

0x0 Non-secure EL1 writes of ICC\_DIR\_EL1 and ICV\_DIR\_EL1 are not trapped to EL2, unless trapped by other mechanisms.



0x1 Non-secure EL1 writes of ICC\_DIR\_EL1 and ICV\_DIR\_EL1 are trapped to EL2.

**TSEI, [13]**

Trap all locally generated SEIs. The value is:

0 Locally generated SEIs do not cause a trap to EL2.

**TALL1, [12]**

Trap all Non-secure EL1 accesses to ICC\_\* and ICV\_\* System registers for Group 1 interrupts to EL2. The possible values are:

0x0 Non-secure EL1 accesses to ICC\_\* and ICV\_\* registers for Group 1 interrupts proceed as normal.

0x1 Non-secure EL1 accesses to ICC\_\* and ICV\_\* registers for Group 1 interrupts trap to EL2.

**TALL0, [11]**

Trap all Non-secure EL1 accesses to ICC\_\* and ICV\_\* System registers for Group 0 interrupts to EL2. The possible values are:

0x0 Non-secure EL1 accesses to ICC\_\* and ICV\_\* registers for Group 0 interrupts proceed as normal.

0x1 Non-secure EL1 accesses to ICC\_\* and ICV\_\* registers for Group 0 interrupts trap to EL2.

**TC, [10]**

Trap all Non-secure EL1 accesses to System registers that are common to Group 0 and Group 1 to EL2. The possible values are:

0x0 Non-secure EL1 accesses to common registers proceed as normal.

0x1 Non-secure EL1 accesses to common registers trap to EL2.

**RES0, [9:8]**

**RES0** Reserved.

**VGrp1DIE, [7]**

VM Group 1 Disabled Interrupt Enable. The possible values are:

0 Maintenance interrupt disabled.

1 Maintenance interrupt signaled when ICH\_VMCR\_EL2.VENG1 is 0.

**VGrp1EIE, [6]**

VM Group 1 Enabled Interrupt Enable. The possible values are:

0 Maintenance interrupt disabled.

1 Maintenance interrupt signaled when ICH\_VMCR\_EL2.VENG1 is 1.

**VGrp0DIE, [5]**

VM Group 0 Disabled Interrupt Enable. The possible values are:

- |   |  |
|---|--|
| 0 | Maintenance interrupt disabled.                              |
| 1 | Maintenance interrupt signaled when ICH_VMCR_EL2.VENG0 is 0. |

**VGrp0EIE, [4]**

VM Group 0 Enabled Interrupt Enable. The possible values are:

- |   |  |
|---|--|
| 0 | Maintenance interrupt disabled.                              |
| 1 | Maintenance interrupt signaled when ICH_VMCR_EL2.VENG0 is 1. |

**NPIE, [3]**

No Pending Interrupt Enable. The possible values are:

- |   |   |
|---|---|
| 0 | Maintenance interrupt disabled.   |
| 1 | Maintenance interrupt signaled while the List registers contain no interrupts in the pending state. |

**LRENPIE, [2]**

List Register Entry Not Present Interrupt Enable. The possible values are:

- |   |  |
|---|--|
| 0 | Maintenance interrupt disabled.                                      |
| 1 | Maintenance interrupt is asserted while the EOIcount field is not 0. |

**UIE, [1]**

Underflow Interrupt Enable. The possible values are:

- |   |  |
|---|--|
| 0 | Maintenance interrupt disabled.  |
| 1 | Maintenance interrupt is asserted if none, or only one, of the List register entries is marked as a valid interrupt. |

**En, [0]**

Enable. The possible values are:

- |   |   |
|---|---|
| 0 | Virtual CPU interface operation disabled. |
| 1 | Virtual CPU interface operation enabled.  |

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

### 3.3.22 ICH\_VMCR\_EL2, Interrupt Controller Virtual Machine Control Register, EL2

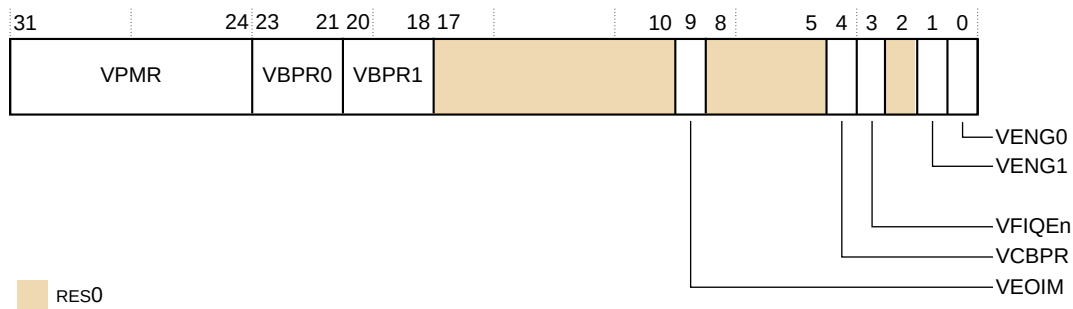
ICH\_VMCR\_EL2 enables the hypervisor to save and restore the virtual machine view of the GIC state.

#### Bit field descriptions

ICH\_VMCR\_EL2 is a 32-bit register and is part of:

- The GIC System registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

**Figure 3-79: ICH\_VMCR\_EL2 bit assignments**



#### VPMR, [31:24]

Virtual Priority Mask.

This field is an alias of ICV\_PMR\_EL1.Priority.

#### VBPR0, [23:21]

Virtual Binary Point Register, Group 0. The minimum value is:

0x2 This field is an alias of ICV\_BPR0\_EL1.BinaryPoint.

#### VBPR1, [20:18]

Virtual Binary Point Register, Group 1. The minimum value is:

0x3 This field is an alias of ICV\_BPR1\_EL1.BinaryPoint.

#### RES0, [17:10]

RES0 Reserved.

#### VEOIM, [9]

Virtual EOI mode. The possible values are:

0x0	ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICV_DIR_EL1 are <b>UNPREDICTABLE</b> .
0x1	ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide priority drop functionality only. ICV_DIR_EL1 provides interrupt deactivation functionality.

This bit is an alias of ICV\_CTLR\_EL1.EOImode.

#### RES0, [8:5]

RES0	Reserved.
------	-----------

#### VCBPR, [4]

Virtual Common Binary Point Register. The possible values are:

0x0	ICV_BPR0_EL1 determines the preemption group for virtual Group 0 interrupts only.
	ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts.
0x1	ICV_BPR0_EL1 determines the preemption group for both virtual Group 0 and virtual Group 1 interrupts.
	Reads of ICV_BPR1_EL1 return ICV_BPR0_EL1 plus one, saturated to 111. Writes to ICV_BPR1_EL1 are <b>IGNORED</b> .

#### VFIQEn, [3]

Virtual FIQ enable. The value is:

0x1	Group 0 virtual interrupts are presented as virtual FIQs.
-----	---

#### RES0, [2]

RES0	Reserved.
------	-----------

#### VENG1, [1]

Virtual Group 1 interrupt enable. The possible values are:

0x0	Virtual Group 1 interrupts are disabled.
0x1	Virtual Group 1 interrupts are enabled.

#### VENG0, [0]

Virtual Group 0 interrupt enable. The possible values are:

0x0	Virtual Group 0 interrupts are disabled.
0x1	Virtual Group 0 interrupts are enabled.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

### 3.3.23 ICH\_VTR\_EL2, Interrupt Controller VGIC Type Register, EL2

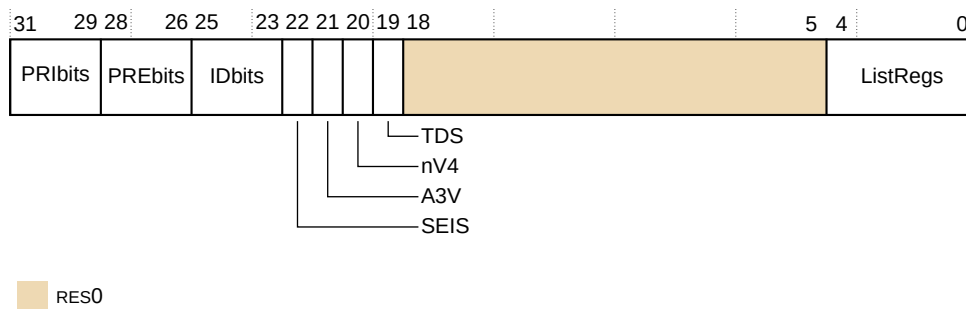
ICH\_VTR\_EL2 reports supported GIC virtualization features.

#### Bit field descriptions

ICH\_VTR\_EL2 is a 32-bit register and is part of:

- The GIC System registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

**Figure 3-80: ICH\_VTR\_EL2 bit assignments**



#### PRIbits, [31:29]

Priority bits. The number of virtual priority bits implemented, minus one.

0x4      Priority implemented is 5-bit.

#### PREbits, [28:26]

The number of virtual preemption bits implemented, minus one. The value is:

0x4      Virtual preemption implemented is 5-bit.

#### IDbits, [25:23]

The number of virtual interrupt identifier bits supported. The value is:

0x0      Virtual interrupt identifier bits that are implemented is 16-bit.

#### SEIS, [22]

SEI Support. The value is:

0x0      The virtual CPU interface logic does not support generation of SEIs.

#### A3V, [21]

Affinity 3 Valid. The value is:

0x1 The virtual CPU interface logic supports nonzero values of Affinity 3 in SGI generation System registers.

#### nV4, [20]

Direct injection of virtual interrupts not supported. The value is:

0x0 The CPU interface logic supports direct injection of virtual interrupts.

#### TDS, [19]

Separate trapping of Non-secure EL1 writes to ICV\_DIR\_EL1 supported. The value is:

0x1 Implementation supports ICH\_HCR\_EL2.TDIR.

#### RES0, [18:5]

RES0 Reserved.

#### ListRegs, [4:0]

0x3 The number of implemented List registers, minus one.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

## 3.4 Advanced SIMD and floating-point registers

This chapter describes the Advanced SIMD and floating-point registers.

### 3.4.1 AArch64 register summary

The core has several Advanced SIMD and floating-point system registers. Each register has a specific purpose, specific usage constraints, configurations, and attributes.

The following table gives a summary of the Cortex®-A65AE core Advanced SIMD and floating-point system registers.

**Table 3-44: AArch64 Advanced SIMD and floating-point system registers**

Name	Type	Reset	Description
FPCR	RW	0x00000000	See <a href="#">3.4.2 FPCR, Floating-point Control Register</a> on page 230.
FPSR	RW	0x00000000	See <a href="#">3.4.3 FPSR, Floating-point Status Register</a> on page 233.

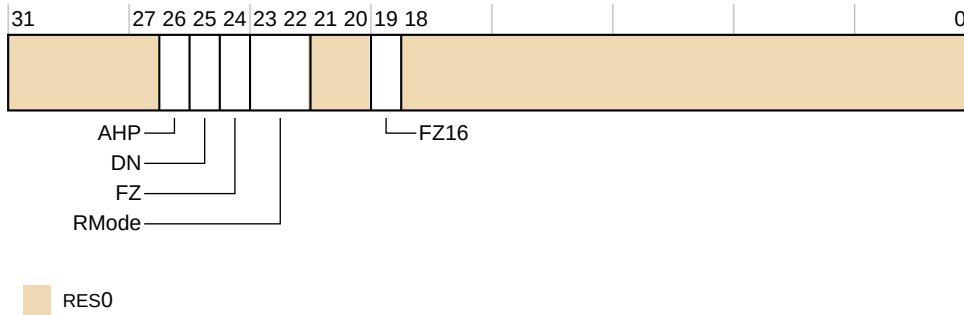
### 3.4.2 FPCR, Floating-point Control Register

The FPCR controls floating-point behavior.

#### Bit field descriptions

FPCR is a 32-bit register.

**Figure 3-81: FPCR bit assignments**



#### RES0, [31:27]

*res0*

Reserved.

#### AHP, [26]

Alternative half-precision control bit. The possible values are:

0

IEEE half-precision format selected. This is the reset value.

1

Alternative half-precision format selected.

#### DN, [25]

Default NaN mode control bit. The possible values are:

0

NaN operands propagate through to the output of a floating-point operation. This is the reset value.

1

Any operation involving one or more NaNs returns the Default NaN.

#### FZ, [24]

Flush-to-zero mode control bit. The possible values are:

**0**

Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. This is the reset value.

**1**

Flush-to-zero mode enabled.

**RMode, [23:22]**

Rounding Mode control field. The encoding of this field is:

0b00	<i>Round to Nearest (RN) mode. This is the reset value.</i>
0b01	<i>Round towards Plus Infinity (RP) mode.</i>
0b10	<i>Round towards Minus Infinity (RM) mode.</i>
0b11	<i>Round towards Zero (RZ) mode.</i>

**RES0, [21:20]****res0**

Reserved.

**FZ16, [19]**

Flush-to-zero mode control bit on half-precision data-processing instructions. The possible values are:

**0**

Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. This is the default value.

**1**

Flush-to-zero mode enabled.

**RES0, [18:0]****res0**

Reserved.

**Configurations**

There are no configurations.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**Usage constraints****Accessing the FPCR**

To access the FPCR:

```
MRS <Xt>, FPCR ; Read FPCR into Xt
MSR FPCR, <Xt> ; Write Xt to FPCR
```

Register access is encoded as follows:



**Table 3-45: FPCR access encoding**

op0	op1	CRn	CRm	op2
11	011	0100	0100	000

**Accessibility**

This register is accessible as follows:

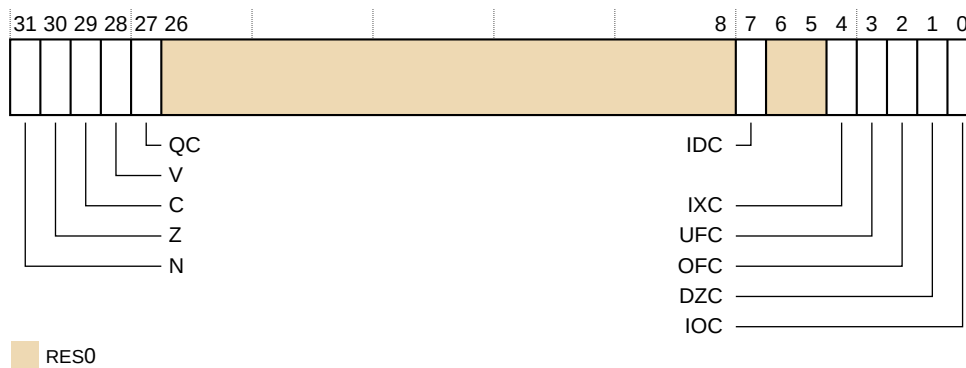
EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
RW	RW	RW	RW	RW	RW

**3.4.3 FPSR, Floating-point Status Register**

The FPSR provides floating-point system status information.

**Bit field descriptions**

FPSR is a 32-bit register.

**Figure 3-82: FPSR bit assignments****N, [31]**

Not in use, RES0. AArch64 floating-point comparisons set the PSTATE.N flag instead.

**Z, [30]**

Not in use, RES0. AArch64 floating-point comparisons set the PSTATE.Z flag instead.

**C, [29]**

Not in use, RES0. AArch64 floating-point comparisons set the PSTATE.C flag instead.

**V, [28]**

Not in use, RES0. AArch64 floating-point comparisons set the PSTATE.V flag instead.

**QC, [27]**

Cumulative saturation bit. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated since a 0 was last written to this bit.

**RES0, [26:8]**

Reserved, *res0*.

**IDC, [7]**

Input Denormal cumulative exception bit. This bit is set to 1 to indicate that the Input Denormal exception has occurred since 0 was last written to this bit.

**RES0, [6:5]**

Reserved, *res0*.

**IXC, [4]**

Inexact cumulative exception bit. This bit is set to 1 to indicate that the Inexact exception has occurred since 0 was last written to this bit.

**UFC, [3]**

Underflow cumulative exception bit. This bit is set to 1 to indicate that the Underflow exception has occurred since 0 was last written to this bit.

**OFC, [2]**

Overflow cumulative exception bit. This bit is set to 1 to indicate that the Overflow exception has occurred since 0 was last written to this bit.

**DZC, [1]**

Division by Zero cumulative exception bit. This bit is set to 1 to indicate that the Division by Zero exception has occurred since 0 was last written to this bit.

**IOC, [0]**

Invalid Operation cumulative exception bit. This bit is set to 1 to indicate that the Invalid Operation exception has occurred since 0 was last written to this bit.

**Configurations**

There are no configurations.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**Usage constraints****Accessing the FPSR**

To access the FPSR:

```
MRS <Xt>, FPSR; Read FPSR into Xt
MSR FPSR, <Xt>; Write Xt to FPSR
```

Register access is encoded as follows:

**Table 3-47: FPSR access encoding**

op0	op1	CRn	CRm	op2
11	011	0100	0100	001

**Accessibility**

This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
RW	RW	RW	RW	RW	RW

## 4 Debug descriptions

This part describes the debug functionality of the Cortex®-A65AE core.

### 4.1 Debug

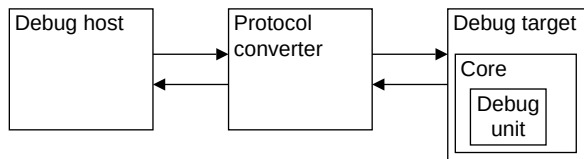
This chapter describes the debug features of the core.

#### 4.1.1 About debug methods

The core is part of a debug system and supports both self-hosted and external debug.

The following figure shows a typical external debug system.

**Figure 4-1: External debug system**



#### Debug host

A computer, for example a personal computer, that is running a software debugger such as the DS-5 Debugger. With the debug host, you can issue high-level commands, such as setting a breakpoint at a certain location or examining 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 using the *Advanced Peripheral Bus* (APB) slave interface. An example of a debug target is a development system with a test chip or a silicon part with a core.

#### Debug unit

Helps debugging software that is running on the core:

- Hardware systems that are based on 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 core.

For self-hosted debug, the debug target runs additional debug monitor software that runs on the Cortex®-A65AE core itself. This way, it does not require expensive interface hardware to connect a second host computer.

## 4.1.2 Debug functional description

This section describes the trace, debug, and test features supported by Cortex®-A65AE. It includes Arm®v8-A Debug, CoreSight™ Debug, and cache Debug.

### Arm®v8-A debug architecture support

The Cortex®-A65AE core supports the Arm®v8-A debug architecture.

The core allows access to the internal debug functionality and registers either through a memory-mapped area on the external AMBA® APBv3 slave port, or by using CP14 system coprocessor operations from software running on the core.

The core implements six hardware breakpoints, four watchpoints, and a *Debug Communications Channel* (DCC). Four of the breakpoints match only against virtual address, the other two breakpoints match against either virtual address or context ID. All watchpoints can be linked to either of the virtual address or context-ID matching breakpoints to allow a memory request to be trapped in a given process context.



#### Arm®v7 debug map support

For backwards compatibility, and to reduce the address space required for the debug map, a 4k page-based memory map is also supported.

---

### CoreSight debug

The Cortex®-A65AE core integrates several CoreSight™ debug related components to aid system debug in conjunction with CoreSight™ SoC.

These components include:

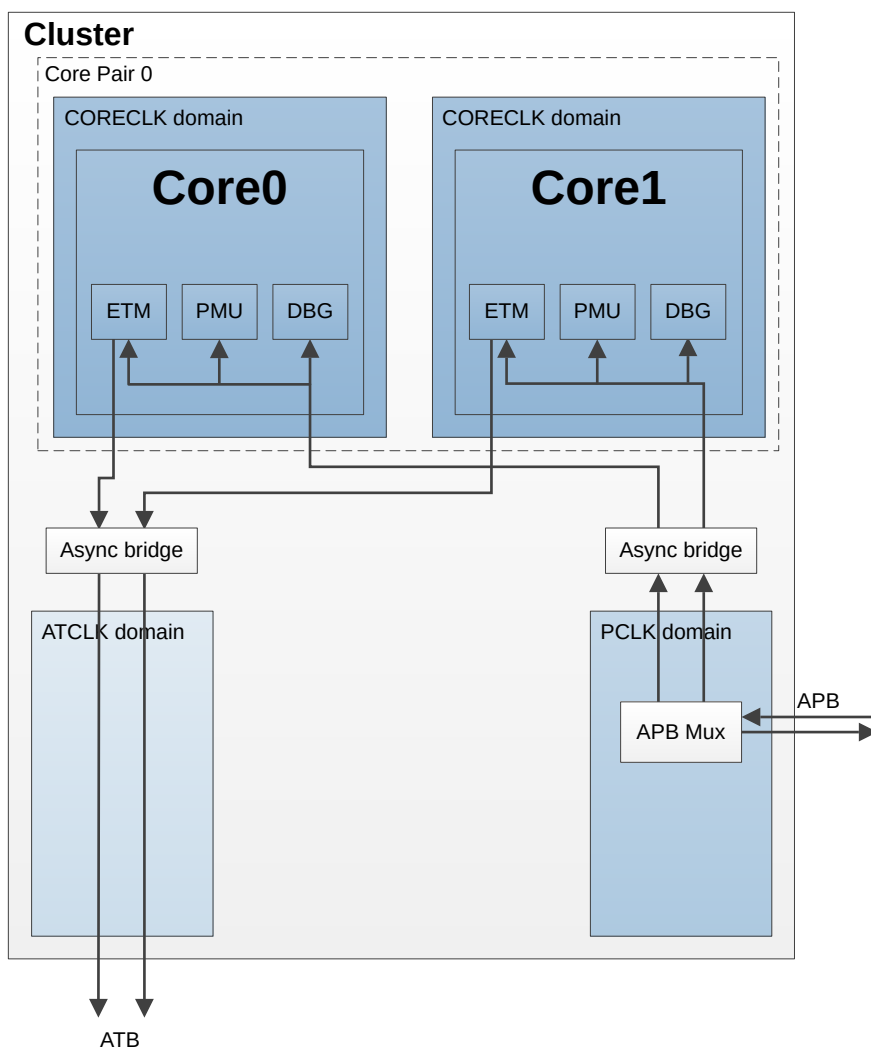
- Per-core *Embedded Trace Macrocell* (ETM).
- Per-core *Cross Trigger Interface* (CTI).
- *Cross Trigger Matrix* (CTM).
- Debug-over-power-down support.

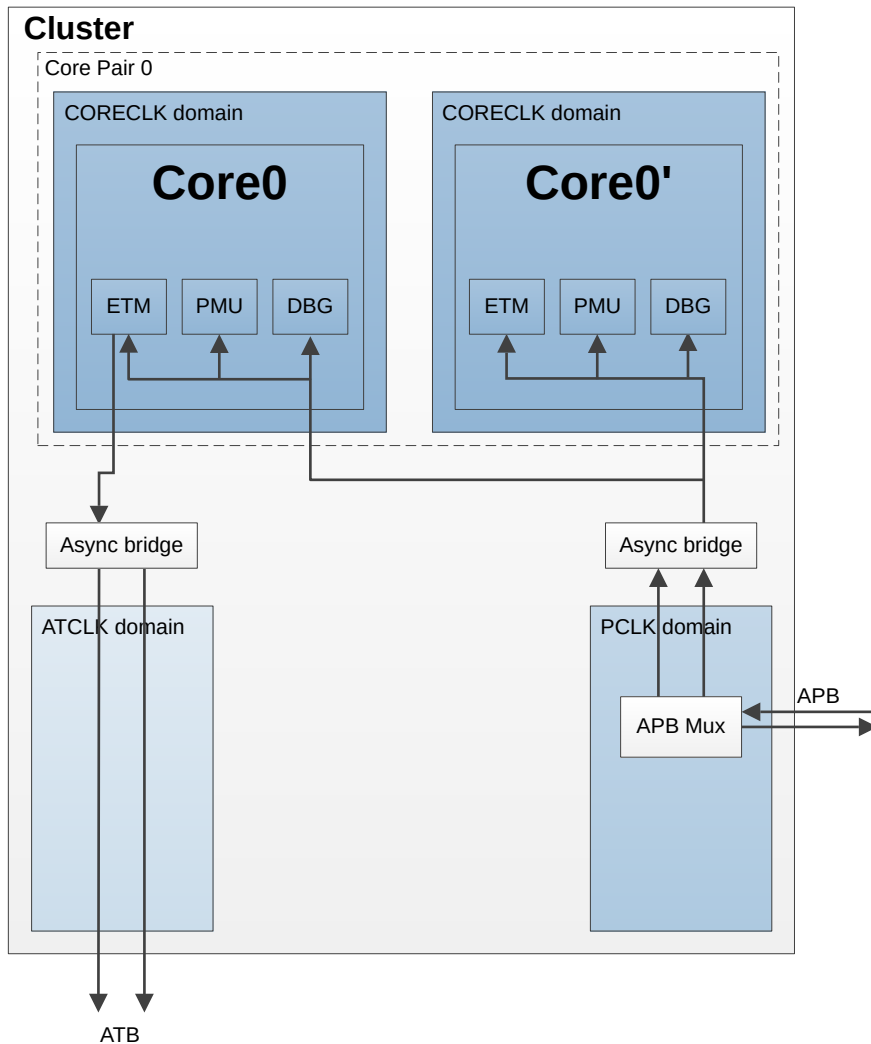
The following figures show the Cortex®-A65AE CoreSight™ debug components in Split-mode and Lock-mode.



The DAP connection is shown for completeness.

**Figure 4-2: Cortex®-A65AE debug components in Split-mode**



**Figure 4-3: Cortex®-A65AE debug components in Lock-mode**

The debug components are split into two groups. Some components are in the cluster itself and the rest are in a separate block named the DebugBlock. It allows you to put the DebugBlock in a separate power domain and place it physically with other CoreSight™ logic in the SoC, rather than close to the cluster.

The connection between the cluster and the DebugBlock consists of a pair of APB interfaces, one in each direction. All debug traffic, except the authentication interface, takes place over this interface as read or write APB transactions. It includes register reads, writes, and CTI triggers.

All debug components are controlled through the primary Debug APB interface on the DebugBlock, and form a standard CoreSight™ interface. Requests on this bus are decoded by the APB decoder before being sent to the appropriate component in the DebugBlock or in the cluster. The per-core CTIs are connected to a CoreSight™ CTM.

Each core contains an ETM, PMU, and debug component that are accessed using the debug APB bus. This block conforms to the Arm®v8-A Debug Architecture Specification.

The core supports debug-over-power-down using modules contained in the DebugBlock that mirror key core information such as core ID. These allow the JTAG scan chain connection to be maintained while the core is powered down.

The ETM in each core outputs trace on a 32-bit AMBA 4 ATBv1.1 interface. There is one interface per core.

### 4.1.3 Debug register interfaces

The core implements the Arm®v8-A Debug architecture and debug events.

They are described in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The Debug architecture defines a set of debug registers. The debug register interfaces provide access to these registers from:

- Software running on the core.
- An external debugger.

#### 4.1.3.1 Core interfaces

System register access allows the core to directly access certain debug registers.

##### **Debug registers**

This function is System register based and memory-mapped. You can access the Debug register map using the APB slave port. The external debug interface enables both external and self-hosted debug agents to access Debug registers. Access to the Debug registers is partitioned as follows:

##### **Performance monitor**

This function is System register based and memory-mapped. You can access the performance monitor registers using the APB slave port.

##### **Trace registers**

This function is memory-mapped.

##### **Related information**

[External debug interface](#) on page 242



### 4.1.3.2 Breakpoints and watchpoints

The core supports six breakpoints, four watchpoints, and a standard *Debug Communications Channel* (DCC) for each thread.

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 virtual address and the other two (BRP 4 and 5) match against either virtual address or context ID, or VMID. 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.

### 4.1.3.3 Effects of resets on Debug registers

The core has the following reset signals that affect the Debug registers:

#### nCPUPORESET

This signal initializes the core processing logic, including the debug, ETM trace unit, breakpoint, watchpoint logic, and performance monitors logic. This maps to a Cold reset that covers reset of the core processing logic and the integrated debug functionality.

#### nCORERESET

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

### 4.1.3.4 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 describes the core response to accesses through the external debug interface.

**Table 4-1: External access conditions to registers**

Name	Condition	Description
Off	EDPRSR.PU is 0	Core power domain is completely off, or in a low-power state where the Core power domain registers cannot be accessed.  If debug power is off, then all external debug and memory-mapped register accesses return an error.
DLK	DoubleLockStatus() == TRUE (EDPRSR.DLK is 1)	OS Double Lock is locked.
OSLK	OSLSR_EL1.OSLK is 1	OS Lock is locked.
EDAD	AllowExternalDebugAccess() ==FALSE	External debug access is disabled. When an error is returned because of an EDAD condition code, and this is the highest priority error condition, EDPRSR.SDAD is set to 1. Otherwise SDAD is unchanged.
Default	-	None of the conditions apply, normal access.

The following table shows an example of external register access condition codes for access to a performance monitor register. To determine the access permission for the register, scan the columns from left to right. Stop at the first column a condition is true, the entry gives the access permission of the register and scanning stops.

**Table 4-2: External register condition code example**

Off	DLK	OSLK	EDAD	Default
-	-	-	-	RO

## 4.1.4 Debug events

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

A core responds to a debug event in one of the following ways:

- Ignores the debug event.
- Takes a debug exception.
- Enters debug state.

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

### Related information

[External debug interface](#) on page 242

[About clocks, resets, and input synchronization](#) on page 31

### 4.1.4.1 Watchpoint debug events

In the Cortex®-A65AE core, watchpoint debug events are always synchronous.

### 4.1.4.2 Debug OS Lock

Debug OS Lock is set by the powerup reset, **nCPUPORESET**.

For normal behavior of debug events and Debug register accesses, Debug OS Lock must be cleared. For more information, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 4.1.5 External debug interface

For information about external debug interface, including debug memory map and debug signals, see the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual*.

## 4.2 Performance Monitor Unit

This chapter describes the *Performance Monitor Unit* (PMU).

### 4.2.1 About the PMU

The Cortex®-A65AE core includes performance monitors that enable you to gather various statistics on the operation of the core and its memory system during runtime. These provide useful information about the behavior of the core that you can use when debugging or profiling code.

The PMU provides six counters. Each counter can count any of the events available in the core. The absolute counts that are recorded might vary because of pipeline effects. This has negligible effect except in cases where the counters are enabled for a very short time.

#### Related information

[PMU events](#) on page 244

### 4.2.2 PMU functional description

This section describes the functionality of the PMU.

The PMU includes the following interfaces and counters:

#### Event interface

Events from all other units from across the design are provided to the PMU.

#### System register and APB interface

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

#### Counters

The PMU has 32-bit counters that increment when they are enabled, based on events, and a 64-bit cycle counter.

#### PMU register interfaces

The Cortex®-A65AE core supports access to the performance monitor registers from the internal System register interface and a memory-mapped interface.

### 4.2.3 External register access permissions to the PMU registers

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

The following table describes the core response to accesses through the external debug and memory-mapped interfaces.

**Table 4-3: External register conditions**

Name	Condition	Description
Off	EDPRSR.PU is 0	Core power domain is completely off, or in a low-power state where the Core power domain registers cannot be accessed.
DLK	EDPRSR.DLK is 1	OS Double Lock is locked.
OSLK	OSLSR_EL1.OSLK is 1	OS Lock is locked.
EPMAID	<code>AllowExternalPMUAccess()</code> == FALSE	External performance monitors access is disabled. When an error is returned because of an EPMAID condition code, and this is the highest priority error condition, EDPRSR.SPMAD is set to 1. Otherwise SPMAD is unchanged.
Default	-	None of the conditions apply, normal access.

The following table shows an example of external register condition codes for access to a performance monitor register. To determine the access permission for the register, scan the columns from left to right. Stop at the first column whose condition is true, the entry gives the register access permission and scanning stops.

**Table 4-4: External register condition code example**

Off	DLK	OSLK	EPMAID	Default
-	-	-	-	RO

## 4.2.4 PMU events

The following table shows the events that are generated and the numbers that the PMU uses to reference the events. The table also shows the bit position of each event on the event bus. Event reference numbers that are not listed are reserved.

**Table 4-5: PMU events**

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x00	-	SW_INCR	Instruction architecturally executed, condition code check pass, software increment.
0x01	[0]	L1I_CACHE_REFILL	<p>Level 1 instruction cache refill.</p> <p>This event counts any instruction fetch which misses in the cache.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Cache maintenance instructions.</li> <li>Non-cacheable accesses.</li> </ul>

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x02	[1]	L1I_TLB_REFILL	<p>Level 1 instruction TLB refill.</p> <p>This event counts any refill of the instruction L1 TLB from the L2 TLB. This includes refills which result in a translation fault.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• TLB maintenance instructions.</li> </ul> <p>This event counts regardless of whether the MMU is enabled.</p>
0x03	[2]	L1D_CACHE_REFILL	<p>Level 1 data cache refill.</p> <p>This event counts any load or store operation or pagewalk access which causes data to be read from outside the L1, including accesses which do not allocate into L1.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• Cache maintenance instructions and prefetches.</li> <li>• Stores of an entire cache line, even if they make a coherency request outside the L1.</li> <li>• Partial cache line writes which do not allocate into the L1 cache.</li> <li>• Non-cacheable accesses.</li> </ul> <p>This event counts the sum of L1D_CACHE_REFILL_RD and L1D_CACHE_REFILL_WR.</p>
0x04	[3]	L1D_CACHE	<p>Level 1 data cache access.</p> <p>This event counts any load or store operation or pagewalk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL event causes this event to count.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• Cache maintenance instructions and prefetches.</li> <li>• Non-cacheable accesses.</li> </ul> <p>This event counts the sum of L1D_CACHE_RD and L1D_CACHE_WR.</p>
0x05	[4]	L1D_TLB_REFILL	<p>Level 1 data TLB refill.</p> <p>This event counts any refill of the data L1 TLB from the L2 TLB. This includes refills which result in a translation fault.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• TLB maintenance instructions.</li> </ul> <p>This event counts regardless of whether the MMU is enabled.</p>

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x06	[5]	LD_RETIRED	<p>Instruction architecturally executed, condition code check pass, load.</p> <p>This event counts all load and prefetch instructions.</p> <p>This includes the Arm®v8.1-A atomic instructions, other than the ST* variants.</p>
0x07	[6]	ST_RETIRED	<p>Instruction architecturally executed, condition code check pass, store.</p> <p>This event counts all store instructions and DC ZVA.</p> <p>This includes all the Arm®v8.1-A atomic instructions.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Store-Exclusive instructions which fail.</li> </ul>
0x08	[7]	INST_RETIRED	<p>Instruction architecturally executed.</p> <p>This event counts all retired instructions, including those that fail their condition check.</p>
0x09	[8]	EXC_TAKEN	Exception taken.
0x0A	[9]	EXC_RETURN	Instruction architecturally executed, condition code check pass, exception return.
0x0B	[10]	CID_WRITE_RETIRED	<p>Instruction architecturally executed, condition code check pass, write to CONTEXTIDR.</p> <p>This event only counts writes via the CONTEXTIDR_EL1 mnemonic in AArch64.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Writes to CONTEXTIDR_EL12 and CONTEXTIDR_EL2.</li> </ul>
0x0C	[11]	PC_WRITE_RETIRED	<p>Instruction architecturally executed, condition code check pass, software change of the PC.</p> <p>This event counts all branches taken and popped from the branch monitor. This excludes exception entries, debug entries, and CCFAIL branches.</p>
0x0D	[12]	BR_IMMED_RETIRED	<p>Instruction architecturally executed, immediate branch.</p> <p>This event counts all branches decoded as immediate branches, taken or not, and popped from the branch monitor. This excludes exception entries, debug entries, and CCFAIL branches.</p>
0x0E	[13]	BR_RETURN_RETIRED	Instruction architecturally executed, condition code check pass, procedure return.
0x10	[15]	BR_MIS_PRED	<p>Mispredicted or not predicted branch speculatively executed.</p> <p>This event counts any predictable branch instruction which is mispredicted either due to dynamic misprediction or because the MMU is off and the branches are statically predicted not taken.</p>
0x11	-	CPU_CYCLES	The counter increments on every cycle.
0x12	[16]	BR_PRED	<p>Predictable branch speculatively executed.</p> <p>This event counts all predictable branches.</p>

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x13	[17]	MEM_ACCESS	<p>Data memory access. This event counts memory accesses due to load or store instructions.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• Instruction fetches.</li> <li>• Cache maintenance instructions.</li> <li>• Translation table walks or prefetches.</li> </ul> <p>This event counts the sum of MEM_ACCESS_RD and MEM_ACCESS_WR.</p>
0x14	[18]	L1I_CACHE	<p>Level 1 instruction cache access.</p> <p>This event counts any instruction fetch which accesses the L1 instruction cache.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• Cache maintenance instructions.</li> <li>• Non-cacheable accesses.</li> </ul>
0x15	[19]	L1D_CACHE_WB	<p>Level 1 data cache Write-Back.</p> <p>This event counts any write back of data from the L1 data cache to L2 or L3. This counts both victim line evictions and snoops, including cache maintenance operations.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• Invalidations which do not result in data being transferred out of the L1.</li> <li>• Full-line writes which write to L2 without writing L1, such as write-streaming mode.</li> </ul>
0x16	[20]	L2D_CACHE	<p>Level 2 unified cache access.</p> <ul style="list-style-type: none"> <li>• If the core is configured with a per-core L2 cache:  This event counts any transaction from L1 which looks up in the L2 cache, and any write-back from the L1 to the L2. Snoops from outside the core and cache maintenance operations are not counted.</li> <li>• If the core is not configured with a per-core L2 cache:  This event counts the cluster cache event, as defined by L3D_CACHE.</li> <li>• If there is neither a per-core cache nor a cluster cache configured, then this event is not implemented.</li> </ul>

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x17	[21]	L2D_CACHE_REFILL	<p>Level 2 unified cache refill.</p> <ul style="list-style-type: none"> <li>If the core is configured with a per-core L2 cache: <p>This event counts any cacheable transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes and prefetches that target this level of cache should not be counted.</p> </li> <li>If the core is not configured with a per-core L2 cache: <p>This event counts the cluster cache event, as defined by L3D_CACHE_REFILL.</p> </li> <li>If there is neither a per-core cache nor a cluster cache configured, then this event is not implemented.</li> </ul>
0x18	[22]	L2D_CACHE_WB	<p>Level 2 unified cache Write-Back.</p> <ul style="list-style-type: none"> <li>If the core is configured with a per-core L2 cache: <p>This event counts any write back of data from the L2 cache to outside the core. This includes snoops to the L2 which return data, regardless of whether they cause an invalidation. Invalidations from the L2 which do not write data outside of the core and snoops which return data from the L1 are not counted.</p> </li> <li>If the core is not configured with a per-core L2 cache, this event is not implemented.</li> </ul>
0x19	[23]	BUS_ACCESS	<p>Bus access.</p> <p>This event counts for every beat of data transferred over the data channels between the core and the SCU. If both read and write data beats are transferred on a given cycle, this event is counted twice on that cycle.</p> <p>This event counts the sum of BUS_ACCESS_RD and BUS_ACCESS_WR.</p>
0x1A	[24]	MEMORY_ERROR	<p>Local memory error.</p> <p>This event counts any correctable or uncorrectable memory error (ECC or parity) in the protected core RAMs.</p>
0x1B	-	INST_SPEC	<p>Operation speculatively executed.</p> <p>This event duplicates INST_RETIRED.</p>
0x1C	[25]	TTBR_WRITE_RETIRED	<p>Instruction architecturally executed, condition code check pass, write to TTBR. This event only counts writes to TTBRO_EL1/TTBR1_EL1 in AArch64.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Accesses to TTBRO_EL12/TTBR1_EL12 or TTBRO_EL2/TTBR1_EL2.</li> </ul>
0x1D	-	BUS_CYCLES	<p>Bus cycles.</p> <p>This event duplicates CPU_CYCLES.</p>
0x1E	-	CHAIN	Odd performance counter chain mode.



Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x20	[26]	L2D_CACHE_ALLOCATE	<p>Level 2 unified cache allocation without refill.</p> <ul style="list-style-type: none"> <li>If the core is configured with a per-core L2 cache: <p>This event counts any full cache line write into the L2 cache which does not cause a linefill, including write-backs from L1 to L2 and full-line writes which do not allocate into L1.</p> </li> <li>If the core is not configured with a per-core L2 cache: <p>This event counts the cluster cache event, as defined by L3D_CACHE_ALLOCATE.</p> </li> <li>If there is neither a per-core cache nor a cluster cache configured, this event is not implemented.</li> </ul>
0x21	[27]	BR_RETIRED	<p>Instruction architecturally executed, branch.</p> <p>This event counts all branches, taken or not, popped from the branch monitor. This excludes exception entries, debug entries, and CCFail branches. In the Cortex®-A65AE core, an ISB is a branch, and even micro architectural ISBs are counted.</p>
0x22	[28]	BR__MIS_PRED_RETIRED	<p>Instruction architecturally executed, mispredicted branch.</p> <p>This event counts any branch counted by BR_RETIRED which is not correctly predicted and causes a pipeline flush.</p>
0x23	[29]	STALL_FRONTEND	<p>No operation issued because of the frontend.</p> <p>The counter counts on any cycle when no operations are issued due to the instruction queue being empty.</p>
0x24	[30]	STALL_BACKEND	<p>No operation issued because of the backend.</p> <p>The counter counts on any cycle when no operations are issued due to a pipeline stall.</p>
0x25	[31]	L1D_TLB	<p>Level 1 data TLB access.</p> <p>This event counts any load or store operation which accesses the data L1 TLB. If both a load and a store are executed on a cycle, this event counts twice. This event counts regardless of whether the MMU is enabled.</p>
0x26	[32]	L1I_TLB	<p>Level 1 instruction TLB access.</p> <p>This event counts any instruction fetch which accesses the instruction L1 TLB. This event counts regardless of whether the MMU is enabled.</p>
0x29	[33]	L3D_CACHE_ALLOCATE	<p>Attributable Level 3 unified cache allocation without refill.</p> <ul style="list-style-type: none"> <li>If the core is configured with a per-core L2 cache and the cluster is configured with an L3 cache: <p>This event counts any full cache line write into the L3 cache which does not cause a linefill, including write-backs from L2 to L3 and full-line writes which do not allocate into L2.</p> </li> <li>If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.</li> </ul>

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x2A	[34]	L3D_CACHE_REFILL	<p>Attributable Level 3 unified cache refill.</p> <ul style="list-style-type: none"> <li>If the core is configured with a per-core L2 cache and the cluster is configured with an L3 cache:</li> </ul> <p>This event counts for any cacheable read transaction returning data from the SCU for which the data source was outside the cluster. Transactions such as ReadUnique are counted here as “read” transactions, even though they can be generated by store instructions. Prefetches and stashes that target the L3 cache are not counted.</p> <ul style="list-style-type: none"> <li>If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.</li> </ul>
0x2B	[35]	L3D_CACHE	<p>Attributable Level 3 unified cache access.</p> <ul style="list-style-type: none"> <li>If the core is configured with a per-core L2 cache and the cluster is configured with an L3 cache:</li> </ul> <p>This event counts for any cacheable read transaction returning data from the SCU, or for any cacheable write to the SCU.</p> <ul style="list-style-type: none"> <li>If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.</li> </ul>
0x2D	[36]	L2D_TLB_REFILL	<p>Attributable Level 2 unified TLB refill.</p> <p>This event counts on any refill of the L2 TLB, caused by either an instruction or data access.</p> <p>This event does not count if the MMU is disabled.</p>
0x2F	[37]	L2D_TLB	<p>Attributable Level 2 unified TLB access.</p> <p>This event counts on any access to the L2 TLB (caused by a refill of any of the L1 TLBs).</p> <p>This event does not count if the MMU is disabled.</p>
0x34	[39]	DTLB_WALK	<p>Access to data TLB that caused a page table walk.</p> <p>This event counts on any data access which causes L2D_TLB_REFILL to count.</p>
0x35	[40]	ITLB_WALK	<p>Access to instruction TLB that caused a page table walk.</p> <p>This event counts on any instruction access which causes L2D_TLB_REFILL to count.</p>

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x36	[41]	LL_CACHE_RD	<p>Last level cache access, read.</p> <ul style="list-style-type: none"> <li>If CPUECTLR.EXTLLC is set:</li> </ul> <p>This event counts any cacheable read transaction which returns a data source of "interconnect cache".</p> <ul style="list-style-type: none"> <li>If CPUECTLR.EXTLLC is not set:</li> </ul> <p>This event is a duplicate of the L*D_CACHE_RD event corresponding to the last level of cache implemented – L3D_CACHE_RD if both per-core L2 and cluster L3 are implemented, L2D_CACHE_RD if only one is implemented, or L1D_CACHE_RD if neither is implemented.</p>
0x37	[42]	LL_CACHE_MISS_RD	<p>Last level cache miss, read.</p> <ul style="list-style-type: none"> <li>If CPUECTLR.EXTLLC is set:</li> </ul> <p>This event counts any cacheable read transaction which returns a data source of "DRAM", "remote" or "inter-cluster peer".</p> <ul style="list-style-type: none"> <li>If CPUECTLR.EXTLLC is not set:</li> </ul> <p>This event is a duplicate of the L*D_CACHE_REFILL_RD event corresponding to the last level of cache implemented – L3D_CACHE_REFILL_RD if both per-core L2 and cluster L3 are implemented, L2D_CACHE_REFILL_RD if only one is implemented, or L1D_CACHE_REFILL_RD if neither is implemented.</p>
0x38	[38]	REMOTE_ACCESS_RD	<p>Access to another socket in a multi-socket system, read.</p> <p>This event counts any read transaction which returns a data source of "remote".</p>
0x40	-	L1D_CACHE_RD	<p>Level 1 data cache access, read.</p> <p>This event counts any load operation or pagewalk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_RD event causes this event to count.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Cache maintenance instructions and prefetches.</li> <li>Non-cacheable accesses.</li> </ul>
0x41	-	L1D_CACHE_WR	<p>Level 1 data cache access, write.</p> <p>This event counts any store operation which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL event causes this event to count.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Cache maintenance instructions and prefetches.</li> <li>Non-cacheable accesses.</li> </ul>

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x42	-	L1D_CACHE_REFILL_RD	<p>Level 1 data cache refill, read.</p> <p>This event counts any load operation or pagewalk access which causes data to be read from outside the L1, including accesses which do not allocate into L1.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Cache maintenance instructions and prefetches.</li> <li>Non-cacheable accesses.</li> </ul>
0x43	-	L1D_CACHE_REFILL_WR	<p>Level 1 data cache refill, write.</p> <p>This event counts any store operation which causes data to be read from outside the L1, including accesses which do not allocate into L1.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Cache maintenance instructions and prefetches.</li> <li>Stores of an entire cache line, even if they make a coherency request outside the L1.</li> <li>Partial cache line writes which do not allocate into the L1 cache.</li> <li>Non-cacheable accesses.</li> </ul>
0x44	-	L1D_CACHE_REFILL_INNER	<p>Level 1 data cache refill, inner.</p> <p>This event counts any L1 D-cache linefill (as counted by L1D_CACHE_REFILL) which hits in the L2 cache, L3 cache, or another core in the cluster.</p>
0x45	-	L1D_CACHE_REFILL_OUTER	<p>Level 1 data cache refill, outer.</p> <p>This event counts any L1 D-cache linefill (as counted by L1D_CACHE_REFILL) which does not hit in the L2 cache, L3 cache, or another core in the cluster, and instead obtains data from outside the cluster.</p>
0x50	-	L2D_CACHE_RD	<p>Level 2 cache access, read.</p> <ul style="list-style-type: none"> <li>If the core is configured with a per-core L2 cache: <p>This event counts any read transaction from L1 which looks up in the L2 cache. Snoops from outside the core are not counted.</p> </li> <li>If the core is configured without a per-core L2 cache: <p>This event counts the cluster cache event, as defined by L3D_CACHE_RD.</p> </li> <li>If there is neither a per-core cache nor a cluster cache configured, this event is not implemented</li> </ul>

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x51	-	L2D_CACHE_WR	<p>Level 2 cache access, write.</p> <ul style="list-style-type: none"> <li>If the core is configured with a per-core L2 cache:</li> </ul> <p>This event counts any write transaction from L1 which looks up in the L2 cache or any write-back from L1 which allocates into the L2 cache. Snoops from outside the core are not counted.</p> <ul style="list-style-type: none"> <li>If the core is configured without a per-core L2 cache:</li> </ul> <p>This event counts the cluster cache event, as defined by L3D_CACHE_WR.</p> <ul style="list-style-type: none"> <li>If there is neither a per-core cache nor a cluster cache configured, this event is not implemented</li> </ul>
0x52	-	L2D_CACHE_REFILL_RD	<p>Level 2 cache refill, read.</p> <ul style="list-style-type: none"> <li>If the core is configured with a per-core L2 cache:</li> </ul> <p>This event counts any cacheable read transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 should not be counted. Transactions such as ReadUnique are counted here as “read” transactions, even though they can be generated by store instructions.</p> <ul style="list-style-type: none"> <li>If the core is configured without a per-core L2 cache:</li> </ul> <p>This event counts the cluster cache event, as defined by L3D_CACHE_REFILL_RD.</p> <ul style="list-style-type: none"> <li>If there is neither a per-core cache nor a cluster cache configured, this event is not implemented</li> </ul>
0x53	-	L2D_CACHE_REFILL_WR	<p>Level 2 cache refill, write.</p> <ul style="list-style-type: none"> <li>If the core is configured with a per-core L2 cache:</li> </ul> <p>This event counts any write transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 should not be counted. Transactions such as ReadUnique are not counted as write transactions.</p> <ul style="list-style-type: none"> <li>If the core is configured without a per-core L2 cache:</li> </ul> <p>This event counts the cluster cache event, as defined by L3D_CACHE_REFILL_WR.</p> <ul style="list-style-type: none"> <li>If there is neither a per-core cache nor a cluster cache configured, this event is not implemented</li> </ul>
0x60	-	BUS_ACCESS_RD	<p>Bus access, read.</p> <p>This event counts for every beat of data transferred over the read data channel between the core and the SCU.</p>
0x61	-	BUS_ACCESS_WR	<p>Bus access, write.</p> <p>This event counts for every beat of data transferred over the write data channel between the core and the SCU.</p>

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x66	-	MEM_ACCESS_RD	<p>Data memory access, read. This event counts memory accesses due to load instructions.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• Instruction fetches.</li> <li>• Cache maintenance instructions.</li> <li>• Translation table walks.</li> <li>• Prefetches.</li> </ul>
0x67	-	MEM_ACCESS_WR	<p>Data memory access, write. This event counts memory accesses due to store instructions.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• Instruction fetches.</li> <li>• Cache maintenance instructions.</li> <li>• Translation table walks.</li> <li>• Prefetches.</li> </ul>
0x68	-	UNALIGNED_LD_SPEC	Unaligned access, read
0x69	-	UNALIGNED_ST_SPEC	Unaligned access, write.
0x6A	-	UNALIGNED_LDST_SPEC	Unaligned access.
0x70	-	LD_SPEC	<p>Operation speculatively executed, load.</p> <p>This event duplicates LD_RETIRE.</p>
0x71	-	ST_SPEC	<p>Operation speculatively executed, store.</p> <p>This event duplicates ST_RETIRE.</p>
0x72	-	LDST_SPEC	<p>Operation speculatively executed, load or store.</p> <p>This event counts the sum of LD_SPEC and ST_SPEC.</p>
0x73	-	DP_SPEC	<p>Operation speculatively executed, integer data processing.</p> <p>This event counts retired integer data-processing instructions.</p>
0x74	-	ASE_SPEC	<p>Operation speculatively executed, Advanced SIMD instruction.</p> <p>This event counts retired Advanced SIMD instructions.</p>
0x75	-	VFP_SPEC	Instruction speculatively executed, Float Point
0x77	-	CRYPTO_SPEC	<p>Operation speculatively executed, Cryptographic instruction.</p> <p>This event counts retired Cryptographic instructions.</p>
0x78	-	BR_IMMED_SPEC	<p>Branch speculatively executed, immediate branch.</p> <p>This event duplicates BR_IMMED_RETIRE.</p>

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x79	-	BR_RETURN_SPEC	Branch speculatively executed, procedure return.  This event duplicates BR_RETURN_RETIRED.
0x7A	-	BR_INDIRECT_SPEC	Branch speculatively executed, indirect branch. This event counts retired indirect branch instructions.
0x7C	-	ISB_SPEC	ISB speculatively executed
0x86	-	EXC_IRQ	Exception taken, IRQ.
0x87	-	EXC_FIQ	Exception taken, FIQ.
0xA0	-	L3D_CACHE_RD	Attributable Level 3 unified cache access, read.  This event counts for any cacheable read transaction returning data from the SCU.  If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.
0xA2	-	L3D_CACHE_REFILL_RD	Attributable Level 3 unified cache refill, read. This event duplicates L3D_CACHE_REFILL.  If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.
0xC0	-	STB_STALL	Merge in the store buffer.
0xC1	-	BIU_EXT_MEM_REQ	External memory request.
0xC2	-	BIU_EXT_MEM_REQ_NC	External memory request to non-cacheable memory.
0xC3	-	L1D_PREF_LINE_FILL	Level 1 data cache refill started due to prefetch. Counts any linefills from the prefetcher which cause an allocation into the L1 D-cache.
0xC4	-	L2D_PREF_LINE_FILL	Level 2 cache refill due to prefetch. <ul style="list-style-type: none"> <li>If the core is configured with a per-core L2 cache:  This event does not count.</li> <li>If the core is configured without a per-core L2 cache:  This event counts the cluster cache event, as defined by L3_PREF_LINE_FILL.</li> <li>If there is neither a per-core cache nor a cluster cache configured, this event is not implemented</li> </ul>
0xC5	-	L3_PREF_LINE_FILL	Level 3 cache refill due to prefetch.  This event counts any linefills from the hardware prefetcher which cause an allocation into the L3 cache.  <b>Note:</b> It might not be possible to distinguish between both hardware and software prefetches and also which prefetches cause an allocation. If so, only hardware prefetches should be counted, regardless of whether they allocate. If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.
0xC6	-	L1D_WS_MODE_ENTER	L1D entering write stream mode.
0xC7	-	L1D_WS_MODE	L1D is in write stream mode.

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0xC8	-	L2D_WS_MODE	Level 2 cache write streaming mode.  This event counts for each cycle where the core is in write-streaming mode and not allocating writes into the L2 cache.
0xC9	-	L3D_WS_MODE	Level 3 cache write streaming mode.  This event counts for each cycle where the core is in write-streaming mode and not allocating writes into the L3 cache.
0xCA	-	TLB_L2TLB_LLWALK_ACCESS	Level 2 TLB last-level walk cache access.  This event does not count if the MMU is disabled.
0xCB	-	TLB_L2TLB_LLWALK_REFILL	Level 2 TLB last-level walk cache refill.  This event does not count if the MMU is disabled.
0xCC	-	TLB_L2TLB_L2WALK_ACCESS	Level 2 TLB level-2 walk cache access.  This event counts accesses to the level-2 walk cache where the last-level walk cache has missed. The event only counts when the translation regime of the pagewalk uses level 2 descriptors. This event does not count if the MMU is disabled.
0xCD	-	TLB_L2TLB_L2WALK_REFILL	Level 2 TLB level-2 walk cache refill.  This event does not count if the MMU is disabled.
0xCE	-	TLB_L2TLB_S2_ACCESS	Level 2 TLB IPA cache access.  This event counts on each access to the IPA cache. <ul style="list-style-type: none"> <li>If a single pagewalk needs to make multiple accesses to the IPA cache, each access is counted.</li> <li>If stage 2 translation is disabled, this event does not count.</li> </ul>
0xCF	-	TLB_L2TLB_S2_REFILL	Level 2 TLB IPA cache refill.  This event counts on each refill of the IPA cache. <ul style="list-style-type: none"> <li>If a single pagewalk needs to make multiple accesses to the IPA cache, each access which causes a refill is counted.</li> <li>If stage 2 translation is disabled, this event does not count.</li> </ul>
0xD0	-	IFU_IC_MISS_WAIT	I-Cache miss on an access from the prefetch block.
0xD1	-	IFU_IUTLB_MISS_WAIT	Counts the cycles spent on a request for Level 2 TLB lookup after a Level 1 IUTLB miss.
0xD2	-	IFU_MICRO_COND_MISPRED	Micro-predictor conditional/direction mispredict, with respect to. if3/if4 predictor.
0xD3	-	IFU_MICRO_CADDR_MISPRED	Micro-predictor address mispredict, with respect to if3/if4 predictor.
0xD4	-	IFU_MICRO_HIT	Micro-predictor hit with immediate redirect.
0xD6	-	IFU_MICRO_NEG_HIT	Micro-predictor negative cache hit.
0xD7	-	IFU_MICRO_CORRECTION	Micro-predictor correction.
0xD8	-	IFU_MICRO_NO_INSTR1	A 2nd instruction could have been pushed but was not because it was non-sequential.
0xD9	-	IFU_MICRO_NO_PRED	Micro-predictor miss.
0xDA	-	IFU_FLUSHED_TLB_MISS	Thread flushed due to TLB miss.
0xDB	-	IFU_FLUSHED_EXCL_TLB_MISS	Thread flushed due to reasons other than TLB miss.
0xDC	-	IFU_ALL_THRDS_RDY	This thread and the other thread both ready for scheduling in if0.
0xDD	-	IFU_WIN_ARB_OTHER_RDY	This thread was arbitrated when the other thread was also ready for scheduling.



Event number	PMU event bus (to trace)	Event mnemonic	Event description
0xDE	-	IFU_WIN_ARB_OTHER_ACT	This thread was arbitrated when the other thread was also active, but not necessarily ready. For example, waiting for I-Cache or TLB.
0xDF	-	IFU_NOT_RDY_FOR_ARB	This thread was not arbitrated because it was not ready for scheduling. For example, due to a cache miss or TLB miss.
0xE0	-	IFU_GOTO_IDLE	The thread moved from an active state to an inactive state (long-term sleep state, causing deallocation of some resources).
0xE1	-	IFU_IC_LOOKUP_UNDER_MISS	I-Cache lookup under miss from other thread.
0xE2	-	IFU_IC_MISS_UNDER_MISS	I-Cache miss under miss from other thread.
0xE3	-	IFU_INSTR_PUSHED	This thread pushed an instruction into the IQ.
0xE4	-	IFU_IC_LF_SP	I-Cache Speculative line fill.
0xE8	-	DPU_BR_COND_RETIRED	Instruction retired, conditional branch.
0xE9	-	DPU_BR_IND_MIS	Instruction retired, indirect branch, mispredicted.
0xEA	-	DPU_BR_COND_MIS	Instruction retired, conditional branch, mispredicted.
0xEB	-	DPU_MEM_ERR_IFU	Memory error (any type) from IFU.
0xEC	-	DPU_MEM_ERR_DCU	Memory error (any type) from DCU.
0xED	-	DPU_MEM_ERR_TLB	Memory error (any type) from TLB.
0xF0	-	L2_L1D_CACHE_WB_UNATT	Unattributable Level 1 data cache write-back. This event occurs when a requestor outside the PE makes a coherency request that results in write-back.
0xF1	-	L2_L2D_CACHE_UNATT	Unattributable Level 2 unified cache access. This event occurs when a requestor outside the PE makes a coherency request that results in level 2 unified cache access.
0xF2	-	L2_L2D_CACHE_RD_UNATT	Unattributable Level 2 unified cache access, read. This event occurs when a requestor outside the PE makes a coherency request that results in level 2 unified cache read access.
0xF3	-	L2_L3D_CACHE_UNATT	Unattributable Level 3 unified cache access. This event occurs when a requestor outside the PE makes a coherency request that results in level 3 unified cache read access.
0xF4	-	L2_L3D_CACHE_RD_UNATT	Unattributable Level 3 unified cache access, read. This event occurs when a requestor outside the PE makes a coherency request that results in level 3 unified cache read access.
0xF5	-	L2_L3D_CACHE_ALLOC_UNATT	Unattributable Level 3 unified cache allocation without refill. This event occurs when a requestor outside the PE makes a coherency request that results in level 3 cache allocate without refill.
0xF6	-	L2_L3D_CACHE_REFILL_UNATT	Unattributable Level 3 unified cache refill. This event occurs when a requestor outside the PE makes a coherency request that results in level 3 cache refill.
0xF7	-	L2D_CACHE_STASH_DROPPED	Level 2 cache stash dropped. This event counts on each stash request received from the interconnect or ACP, that is targeting L2 and gets dropped due to lack of buffer space to hold the request.

## L2 and L3 cache events (L2D\_CACHE\*, L3D\_CACHE\*)

The behavior of these events depends on the configuration of the core.

If the private L2 cache is present, the L2D\_CACHE\* events count the activity in the private L2 cache, and the L3D\_CACHE\* events count the activity in the DSU-AE L3 cache (if present).

If the private L2 cache is not present but the DSU-AE L3 cache is present, the L2D\_CACHE\* events count activity in the DSU-AE L3 cache and the L3D\_CACHE\* events do not count. The L2D\_CACHE\_WB, L2D\_CACHE\_WR and L2D\_CACHE\_REFILL\_WR events do not count in this configuration.

If neither the private L2 cache nor the DSU-AE L3 cache are present, neither the L2D\_CACHE\* or L3D\_CACHE\* events will count.

### Last Level cache events (LL\_CACHE\_\*)

The behavior of these events depends on the configuration of the core and the value of the CPUECTLR\_EL1.EXTLLC bit.

#### If the EXTLLC bit is 0:

These events count activity in the last level of data cache implemented in the core. This is the DSU-AE L3 cache if it is present, else the private L2 cache if it is present, otherwise the L1 data cache.

#### If the EXTLLC bit is 1:

These events count activity in a last level cache outside the core (if present). These events may not count in all implementations.

## 4.2.5 PMU interrupts

The Cortex®-A65AE core asserts the **nPMUIRQ** signal when the PMU generates an interrupt.

You can route this signal to an external interrupt controller for prioritization and masking. This is the only mechanism that signals this interrupt to the core.

This interrupt is also driven as a trigger input to the CTI. See the *Arm® DynamIQ™ Shared Unit AE Technical Reference Manual* for more information.

## 4.2.6 Exporting PMU events

Some of the PMU events are exported to the ETM trace unit to be monitored.



The **PMUEVENT** bus is not exported to external components. This is because the event bus cannot safely cross an asynchronous boundary when events can be generated on every cycle.

---

## 4.3 Embedded Trace Macrocell

This chapter describes the *Embedded Trace Macrocell* (ETM) for the Cortex®-A65AE core.

### 4.3.1 About the ETM

This module performs real-time instruction flow tracing that complies with the ETM architecture, ETMv4.2. As a CoreSight component, it is part of the Arm real-time debug solution.

### 4.3.2 ETM trace unit generation options and resources

The following table shows the trace generation options that are implemented in the Cortex®-A65AE ETM trace unit.

**Table 4-6: ETM trace unit generation options implemented**

Description	Configuration
Instruction address size in bytes	8
Data address size in bytes	0
Data value size in bytes	0
Virtual Machine ID size in bytes	4
Context ID size in bytes	4
Support for conditional instruction tracing	Not implemented
Support for tracing of data	Not implemented
Support for tracing of load and store instructions as PO elements	Not implemented
Support for cycle counting in the instruction trace	Implemented
Support for branch broadcast tracing	Implemented
Number of events 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	Implemented
Support for overflow avoidance	Not implemented
Support for using CONTEXTIDR_EL2 in VMID comparator	Implemented

The following table shows the resources that are implemented in the Cortex®-A65AE ETM trace unit.

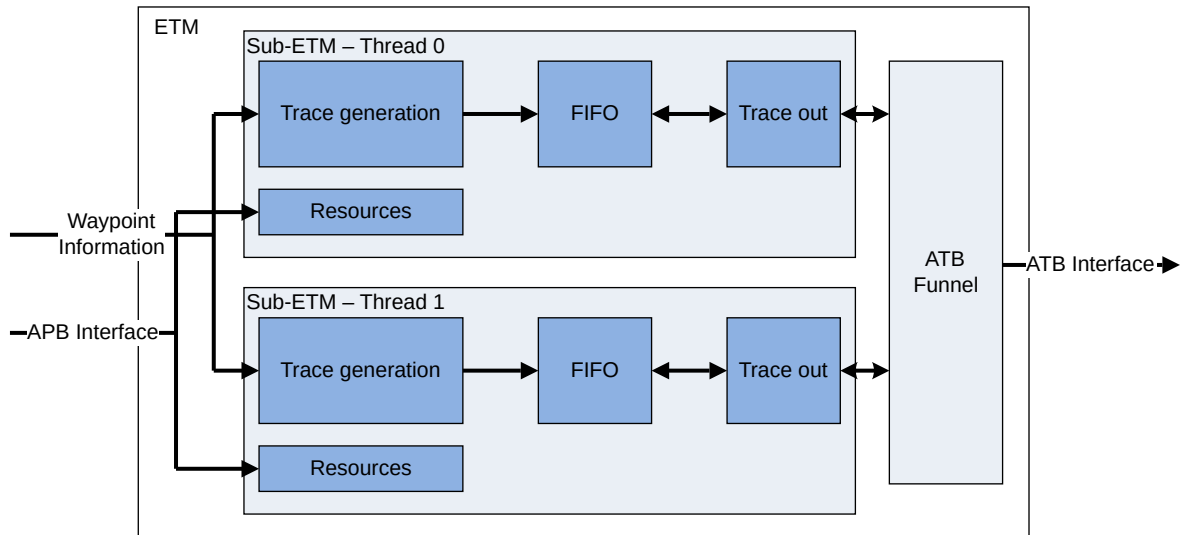
**Table 4-7: ETM trace unit resources implemented**

Description	Configuration
Number of resource selection pairs implemented	8
Number of external input selectors implemented	4
Number of external inputs implemented	47, 4 CTI + 43 PMU  External input bits [0:3] CTI  External input bits [4:46] PMU
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

### 4.3.3 ETM trace unit functional description

This section describes the functionality of each ETM trace unit.

The following figure shows the main functional blocks of each ETM trace unit.

**Figure 4-4: ETM functional blocks****Core interface**

This block monitors the behavior of the core and generates PO elements that are essentially executed branches and exceptions traced in program order.

**Trace generation**

The trace generation block generates various trace packets based on PO elements.

**Filtering and triggering resources**

You can limit the amount of trace data generated by each ETM through the process of filtering.

For example, generating trace only in a certain address range. More complicated logic analyzer style filtering options are also available.

The ETM trace unit can also generate a trigger that is a signal to the Trace Capture Device to stop capturing trace.

**FIFO**

The trace generated by each ETM trace unit is in a highly-compressed form.

The FIFO enables trace bursts to be flattened out. When the FIFO becomes full, the FIFO signals an overflow. The trace generation logic does not generate any new trace until the FIFO is emptied. This causes a gap in the trace when viewed in the debugger.

**Trace out**

Trace from FIFO is output on the AMBA ATB interface.

**ATB funnel**

Combines the ATB interfaces from each ETM into one ATB interface.

### 4.3.4 Resetting the ETM

The reset for each ETM trace unit is the same as a Cold reset for the core.

Each ETM trace unit is not reset when Warm reset is applied to the core so that tracing through Warm core reset is possible.

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



The ETM trace units cannot be reset individually, they can only be reset together.

---

### 4.3.5 Programming and reading ETM trace unit registers

You program and read the ETM trace unit registers using the Debug APB interface.

The core does not have to be in debug state when you program the ETM trace unit registers.

When you are programming the ETM trace unit registers, you must enable all the changes at the same time. Otherwise, if you program the counter, it might start to count based on incorrect events before the correct setup is in place for the trigger condition.

To disable the ETM trace unit, use the TRCPRGCTLR.EN bit.

#### Figure 4-5: Programming ETM trace unit registers

#### Related information

[TRCPRGCTLR, Programming Control Register](#) on page 372

### 4.3.6 ETM trace unit register interfaces

The Cortex®-A65AE core supports only memory-mapped interface to trace registers.

See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4* for information on the behaviors on register accesses for different trace unit states and the different access mechanisms.



Each thread can only access its own ETM trace unit.

---

## Related information

[External debug interface](#) on page 242

### 4.3.7 Interaction with the PMU and Debug

This section describes the interaction with the PMU and the effect of debug double lock on trace register access.

#### Interaction with the PMU

The Cortex®-A65AE core includes a PMU that enables events, such as cache misses and instructions executed, to be counted over a period of time.

The PMU and ETM trace unit function together.

#### Use of PMU events by the ETM trace unit

The PMU architectural events described in [4.2.4 PMU events](#) on page 244 are available to the ETM trace unit through the extended input facility.

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

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

#### Attributability

If an event is attributable to a given thread, then that event is only sent to the ETM associated with that thread. If an event is not attributable to a given thread, then that event is sent to both ETMs.

## Related information

[Performance Monitor Unit](#) on page 242

## 5 Debug registers

This part describes the debug registers of the Cortex®-A65AE core.

### 5.1 AArch64 debug registers

This chapter describes the debug registers in the AArch64 Execution state and shows examples of how to use them.

#### 5.1.1 AArch64 Debug register summary

This section summarizes the debug control registers that are accessible.

These registers, listed in the following table, are accessed by the `MRS` and `MSR` instructions in the order of Op0, CRn, Op1, CRm, Op2.

See [5.2.1 Memory-mapped Debug register summary](#) on page 275 for a complete list of registers accessible from the external debug interface. The 64-bit registers cover two addresses on the external memory interface. For those registers that are not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**Table 5-1: AArch64 debug register summary**

Name	Type	Reset	Width	Description
OSDTRRX_EL1	RW	0x00000000	32	Debug Data Transfer Register, Receive, External View
DBGBVR0_EL1	RW	-	64	Debug Breakpoint Value Register 0
DBGBCR0_EL1	RW	UNK	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
DBGWVR0_EL1	RW	-	64	Debug Watchpoint Value Register 0
DBGWCR0_EL1	RW	UNK	32	<a href="#">5.1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</a> on page 269
DBGBVR1_EL1	RW	-	64	Debug Breakpoint Value Register 1
DBGBCR1_EL1	RW	UNK	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
DBGWVR1_EL1	RW	-	64	Debug Watchpoint Value Register 1
DBGWCR1_EL1	RW	UNK	32	<a href="#">5.1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</a> on page 269
MDCCINT_EL1	RW	0x00000000	32	Monitor Debug Comms Channel Interrupt Enable Register
MDSCR_EL1	RW	-	32	<a href="#">5.1.5 MDSCR_EL1, Monitor Debug System Control Register, EL1</a> on page 272
DBGBVR2_EL1	RW	-	64	Debug Breakpoint Value Register 2
DBGBCR2_EL1	RW	UNK	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
DBGWVR2_EL1	RW	-	64	Debug Watchpoint Value Register 2
DBGWCR2_EL1	RW	UNK	32	<a href="#">5.1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</a> on page 269
OSDTRTX_EL1	RW	-	32	Debug Data Transfer Register, Transmit, External View
DBGBVR3_EL1	RW	-	64	Debug Breakpoint Value Register 3
DBGBCR3_EL1	RW	UNK	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265



Name	Type	Reset	Width	Description
DBGWVR3_EL1	RW	-	64	Debug Watchpoint Value Register 3
DBGWCR3_EL1	RW	UNK	32	<a href="#">5.1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</a> on page 269
DBGBVR4_EL1	RW	-	64	Debug Breakpoint Value Register 4
DBGBCR4_EL1	RW	UNK	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
DBGBVR5_EL1	RW	-	64	Debug Breakpoint Value Register 5
DBGBCR5_EL1	RW	UNK	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
OSECCR_EL1	RW	0x00000000	32	Debug OS Lock Exception Catch Register
MDCCSR_ELO	RO	0x00000000	32	Monitor Debug Comms Channel Status Register
DBGDTR_ELO	RW	0x00000000	64	Debug Data Transfer Register, half-duplex
DBGDTRTX_ELO	WO	0x00000000	32	Debug Data Transfer Register, Transmit, Internal View
DBGDTRRX_ELO	RO	0x00000000	32	Debug Data Transfer Register, Receive, Internal View
MDRAR_EL1	RO	-	64	Debug ROM Address Register. This register is reserved, RES0
OSLAR_EL1	WO	-	32	Debug OS Lock Access Register
OSLSR_EL1	RO	0x0000000A	32	Debug OS Lock Status Register
OSDLR_EL1	RW	0x00000000	32	Debug OS Double Lock Register
DBGPRCR_EL1	RW	-	32	Debug Power/Reset Control Register
DBGCLAIMSET_EL1	RW	0x000000FF	32	<a href="#">5.1.3 DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1</a> on page 269
DBGCLAIMCLR_EL1	RW	0x00000000	32	Debug Claim Tag Clear Register
DBGAUTHSTATUS_EL1	RO	0x000000AA	32	Debug Authentication Status Register

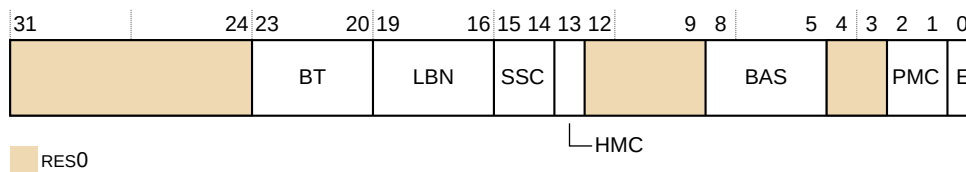
## 5.1.2 DBGBCRn\_EL1, Debug Breakpoint Control Registers, EL1

The DBGBCRn\_EL1 registers hold control information for a breakpoint. Each DBGBVRn\_EL1 is associated with a DBGBCRn\_EL1 to form a *Breakpoint Register Pair* (BRP). The range of *n* for DBGBCRn\_EL1 registers is 0 to 5.

### Bit field descriptions

The DBGBCRn\_EL1 registers are 32-bit registers.

**Figure 5-1: DBGBCRn\_EL1 bit assignments**



**RES0, [31:24]**

*res0*

Reserved.

**BT, [23:20]**

Breakpoint Type. This field controls the behavior of Breakpoint debug event generation. This includes the meaning of the value held in the associated DBGBVR $n$ \_EL1, indicating whether it is an instruction address match or mismatch, or a Context match. It also controls whether the breakpoint is linked to another breakpoint. The possible values are:

**0b0000**

Unlinked instruction address match.

**0b0001**

Linked instruction address match.

**0b0010**

Unlinked Context ID match.

**0b0011**

Linked Context ID match.

**0b0100**

Unlinked instruction address mismatch.

**0b0101**

Linked instruction address mismatch.

**0b0110**

Unlinked CONTEXTIDR\_EL1 match.

**0b0111**

Linked CONTEXTIDR\_EL1 match.

**0b1000**

Unlinked VMID match.

**0b1001**

Linked VMID match.

**0b1010**

Unlinked VMID + Context ID match.

**0b1011**

Linked VMID + Context ID match.

**0b1100**

Unlinked CONTEXTIDR\_EL2 match.

**0b1101**

Linked CONTEXTIDR\_EL2 match.

**0b1110**

Unlinked Full Context ID match.

**0b1111**

Linked Full Context ID match.

The field breakdown is:

- BT[3:1]: Base type. If the breakpoint is not context-aware, these bits are **RES0**. Otherwise, the possible values are:

0b000	Match address. DBGBVR <sub>n</sub> _EL1 is the address of an instruction.
0b001	Match context ID. DBGBVR <sub>n</sub> _EL1[31:0] is a context ID.
0b010	Address mismatch. Mismatch address. Behaves as type 0b000 if either: <ul style="list-style-type: none"> <li>◦ In an AArch64 translation regime.</li> <li>◦ Halting debug-mode is enabled and halting is allowed.</li> </ul> Otherwise, DBGBVR <sub>n</sub> _EL1 is the address of an instruction to be stepped.
0b011	Match CONTEXTIDR_EL1. DBGBVR <sub>n</sub> _EL1[31:0] is a context ID.
0b100	Match VMID. DBGBVR <sub>n</sub> _EL1[47:32] is a VMID.
0b101	Match VMID and CONTEXTIDR_EL1. DBGBVR <sub>n</sub> _EL1[31:0] is a context ID, and DBGBVR <sub>n</sub> _EL1[47:32] is a VMID.
0b110	Match CONTEXTIDR_EL2. DBGBVR <sub>n</sub> _EL1[63:32] is a context ID.
0b111	Match CONTEXTIDR_EL1 and CONTEXTIDR_EL2. DBGBVR <sub>n</sub> _EL1[31:0] and DBGBVR <sub>n</sub> _EL1[63:32] are Context IDs.

- BT[0]: Enable linking.

#### LBN, [19:16]

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

#### SSC, [15:14]

Security State Control. Determines the Security states under which a breakpoint debug event for breakpoint *n* is generated.

This field must be interpreted with the *Higher Mode Control* (HMC), and *Privileged Mode Control* (PMC), fields to determine the mode and Security states that can be tested.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for possible values of the HMC and PMC fields.

#### HMC, [13]

Hyp Mode Control bit. Determines the debug perspective for deciding when a breakpoint debug event for breakpoint *n* is generated.

This bit must be interpreted with the SSC and PMC fields to determine the mode and Security states that can be tested.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for possible values of the SSC and PMC fields.

**RES0, [12:9]****res0**

Reserved.

**BAS, [8:5]**

Byte Address Select. Defines which halfwords a regular breakpoint matches, regardless of the instruction set and Execution state. A debugger must program this field as follows:

**0xF**Match the A64 instruction at DBGBVR<sub>n</sub>\_EL1, or context match.

All other values are reserved.

The Arm®v8-A architecture does not support direct execution of Java bytecodes. BAS[3] and BAS[1] ignore writes and on reads return the values of BAS[2] and BAS[0] respectively.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information on how the BAS field is interpreted by hardware.

**RES0, [4:3]****res0**

Reserved.

**PMC, [2:1]**

Privileged Mode Control. Determines the Exception level or levels that a breakpoint debug event for breakpoint *n* is generated.

This field must be interpreted with the SSC and HMC fields to determine the mode and Security states that can be tested.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for possible values of the SSC and HMC fields.

Bits[2:1] have no effect for accesses made in Hyp mode.

**E, [0]**

Enable breakpoint. This bit enables the BRP:

**0**

BRP disabled.

**1**

BRP enabled.

A BRP never generates a breakpoint debug event when it is disabled.

The value of DBGBCR<sub>n</sub>\_EL1.E is **UNKNOWN** on reset. A debugger must ensure that DBGBCR<sub>n</sub>\_EL1.E has a defined value before it enables debug.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

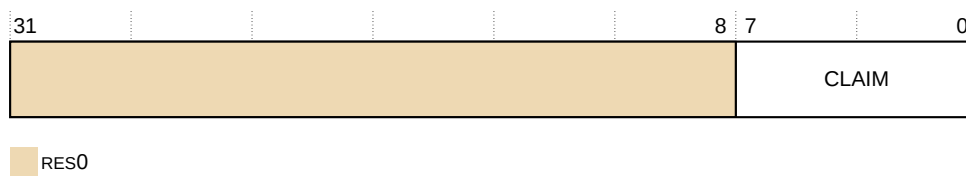
### 5.1.3 DBGCLAIMSET\_EL1, Debug Claim Tag Set Register, EL1

The DBGCLAIMSET\_EL1 is used by software to set CLAIM bits to 1.

#### Bit field descriptions

The DBGCLAIMSET\_EL1 is a 32-bit register.

**Figure 5-2: DBGCLAIMSET\_EL1 bit assignments**



#### RES0, [31:8]

*res0*

Reserved.

#### CLAIM, [7:0]

Claim set bits.

Writing a 1 to one of these bits sets the corresponding CLAIM bit to 1. This is an indirect write to the CLAIM bits.

A single write operation can set multiple bits to 1. Writing 0 to one of these bits has no effect.

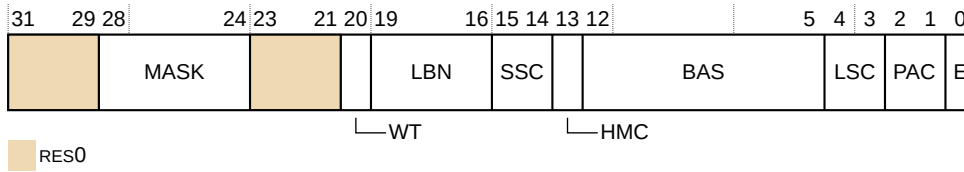
Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 5.1.4 DBGWCRn\_EL1, Debug Watchpoint Control Registers, EL1

The DBGWCRn\_EL1 registers hold control information for a watchpoint. Each DBGWCRn\_EL1 is associated with a DBGWVRn\_EL1 to form a *Watchpoint Register Pair* (WRP). The range of *n* for DBGWCRn\_EL1 registers is 0 to 3.

#### Bit field descriptions

The DBGWCRn\_EL1 registers are 32-bit registers.

**Figure 5-3: DBGWCRn\_EL1 bit assignments****RES0, [31:29]****RES0**

Reserved.

**MASK, [28:24]**

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

0b00000 No mask.  
 0b00001 Reserved.  
 0b00010 Reserved.

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

**RES0, [23:21]****RES0**

Reserved.

**WT, [20]**

Watchpoint type. Possible values are:

**0b0**

Unlinked data address match.

**0b1**

Linked data address match.

On Cold reset, the field reset value is architecturally **UNKNOWN**.

**LBN, [19:16]**

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

On Cold reset, the field reset value is architecturally **UNKNOWN**.

**SSC, [15:14]**

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

On Cold reset, the field reset value is architecturally **UNKNOWN**.

**HMC, [13]**

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

On Cold reset, the field reset value is architecturally **UNKNOWN**.

**BAS, [12:5]**

Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVRn\_EL1 is being watched. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information.

**LSC, [4:3]**

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

**0b01**

Match instructions that load from a watchpoint address.

**0b10**

Match instructions that store to a watchpoint address.

**0b11**

Match instructions that load from or store to a watchpoint address.

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

**IGNORED**

On Cold reset, the field reset value is architecturally **UNKNOWN**.

**PAC, [2:1]**

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

On Cold reset, the field reset value is architecturally **UNKNOWN**.

**E, [0]**

Enable watchpoint n. Possible values are:

**0b0**

Watchpoint disabled.

**0b1**

Watchpoint enabled.

On Cold reset, the field reset value is architecturally **UNKNOWN**.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

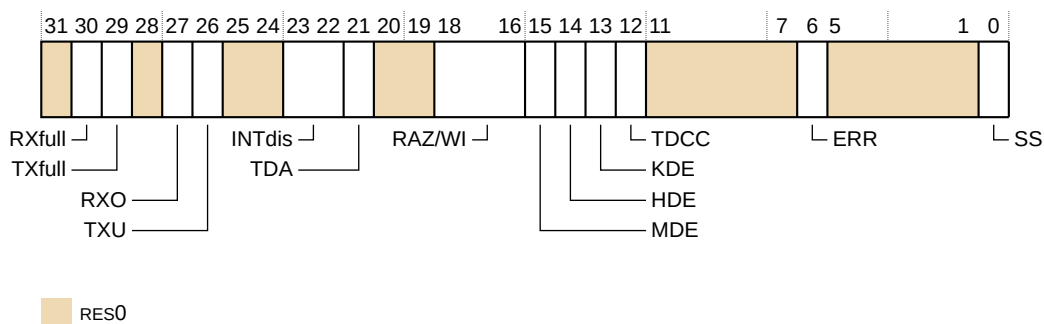
### 5.1.5 MDSCR\_EL1, Monitor Debug System Control Register, EL1

The MDSCR\_EL1 main control register for the debug implementation.

#### Bit field descriptions

MDSCR\_EL1 is a 32-bit register, and is part of the Debug registers functional group.

**Figure 5-4: MDSCR\_EL1 bit assignments**



#### RES0, [31]

**RES0** Reserved.

#### RXfull, [30]

Used for save/restore of EDSCR.RXfull

- When OSLSR\_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.
- When OSLSR\_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

#### TXfull, [29]

Used for save/restore of EDSCR.RXfull

- When OSLSR\_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.
- When OSLSR\_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

#### RES0, [28]

**RES0** Reserved.

#### RXO, [27]

Used for save/restore of EDSCR.RXO.



- When OSLSR\_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as **UNKNOWN** and use an SBZP policy for writes.
- When OSLSR\_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

**TXU, [26]**

Used for save/restore of EDSCR.TXU.

- When OSLSR\_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as **UNKNOWN** and use an SBZP policy for writes.
- When OSLSR\_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

**RES0, [25:24]**

**RES0** Reserved.

**INTdis, [23:22]**

Used for save/restore of EDSCR.INTdis.

- When OSLSR\_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as **UNKNOWN** and use an SBZP policy for writes.
- When OSLSR\_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

**TDA, [21]**

Used for save/restore of EDSCR.TDA.

- When OSLSR\_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as **UNKNOWN** and use an SBZP policy for writes.
- When OSLSR\_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

**RES0, [20:19]**

**RES0** Reserved.

**RAZ/WI, [18:16]**

Reserved, RAZ/WI. Hardware must implement this as RAZ/WI. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

**MDE, [15]**

Monitor debug events. Enable Breakpoint, Watchpoint, and Vector catch debug exceptions.

0	Breakpoint, Watchpoint, and Vector catch debug exceptions disabled.
1	Breakpoint, Watchpoint, and Vector catch debug exceptions enabled.

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

**HDE, [14]**

Used for save/restore of EDSCR.HDE.

- When OSLSR\_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as **UNKNOWN** and use an SBZP policy for writes.

- When OSLSR\_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

**KDE, [13]**

Local (kernel) debug enable. Enable Software debug events within EL<sub>D</sub>. Permitted values are:

- |   |   |
|---|---|
| 0 | Software debug events, other than Software breakpoint instructions, disabled within EL <sub>D</sub> . |
| 1 | Software debug events enabled within EL <sub>D</sub> .  |

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

**TDCC, [12]**

Traps EL0 accesses to the DCC registers to EL1. Permitted values are:

- |   |   |
|---|---|
| 0 | This control does not cause any instructions to be trapped.   |
| 1 | EL0 accesses to the MDCCSR_ELO, DBGDTR_ELO, DBGDTRTX_ELO, and DBGDTRRX_ELO registers are trapped to EL1. Trap of AArch64 accesses to DBGDTR_ELO, DBGDTRRX_ELO, and DBGDTRTX_ELO are ignored in Debug state. |

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

**RES0, [11:7]**

**RES0** Reserved.

**ERR, [6]**

Used for save/restore of EDSCR.ERR.

- When OSLSR\_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as **UNKNOWN** and use an SBZP policy for writes.
- When OSLSR\_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

**RES0, [5:1]**

**RES0** Reserved.

**SS, [0]**

Software step control bit. If EL<sub>D</sub> is used, enables Software step. Permitted values are:

- |   |                            |
|---|----------------------------|
| 0 | Software step is disabled. |
| 1 | Software step is enabled.  |

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

## Configurations

This register is in the Warm reset domain. Some or all RW fields of this register have defined reset values. On a Warm or Cold reset RW fields in this register reset to architecturally **UNKNOWN** values.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 5.2 Memory-mapped Debug registers

This chapter describes the memory-mapped Debug registers and shows examples of how to use them.

### 5.2.1 Memory-mapped Debug register summary

The following table shows the offset address for the registers that are accessible from the external debug interface.

For those registers that are not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**Table 5-2: Memory-mapped debug register summary**

Offset	Name	Type	Width	Description
0x000–0x01C	-	-	-	Reserved
0x020	EDESR	RW	32	External Debug Event Status Register
0x024	EDECR	RW	32	External Debug Execution Control Register
0x028–0x02C	-	-	-	Reserved
0x030	EDWAR[31:0]	RO	64	External Debug Watchpoint Address Register
0x034	EDWAR[63:32]			
0x038–0x07C	-	-	-	Reserved
0x080	DBGDTRRX_ELO	RW	32	Debug Data Transfer Register, Receive
0x084	EDITR	WO	32	External Debug Instruction Transfer Register
0x088	EDSCR	RW	32	External Debug Status and Control Register
0x08C	DBGDTRTX_ELO	WO	32	Debug Data Transfer Register, Transmit
0x090	EDRCR	WO	32	<a href="#">5.2.17 EDRCR, External Debug Reserve Control Register</a> on page 290
0x094	-	RW	32	Reserved
0x098	EDECCR	RW	32	External Debug Exception Catch Control Register
0x09C	-	-	-	Reserved
0x0A0	-	-	-	Reserved
0x0A4	-	-	-	Reserved
0x0A8	-	-	-	Reserved

Offset	Name	Type	Width	Description
0x0AC	-	-	-	Reserved
0x0B0–0x2FC	-	-	-	Reserved
0x300	OSLAR_EL1	WO	32	OS Lock Access Register
0x304–0x30C	-	-	-	Reserved
0x310	EDPRCR	RW	32	External Debug Power/Reset Control Register
0x314	EDPRSR	RO	32	External Debug Processor Status Register
0x318–0x3FC	-	-	-	Reserved
0x400	DBGBVR0_EL1[31:0]	RW	64	Debug Breakpoint Value Register 0
0x404	DBGBVR0_EL1[63:32]			
0x408	DBGBCR0_EL1	RW	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
0x40C	-	-	-	Reserved
0x410	DBGBVR1_EL1[31:0]	RW	64	Debug Breakpoint Value Register 1
0x414	DBGBVR1_EL1[63:32]			
0x418	DBGBCR1_EL1	RW	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
0x41C	-	-	-	Reserved
0x420	DBGBVR2_EL1[31:0]	RW	64	Debug Breakpoint Value Register 2
0x424	DBGBVR2_EL1[63:32]			
0x428	DBGBCR2_EL1	RW	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
0x42C	-	-	-	Reserved
0x430	DBGBVR3_EL1[31:0]	RW	64	Debug Breakpoint Value Register 3
0x434	DBGBVR3_EL1[63:32]			
0x438	DBGBCR3_EL1	RW	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
0x43C	-	-	-	Reserved
0x440	DBGBVR4_EL1[31:0]	RW	64	Debug Breakpoint Value Register 4
0x444	DBGBVR4_EL1[63:32]			
0x448	DBGBCR4_EL1	RW	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
0x44C	-	-	-	Reserved
0x450	DBGBVR5_EL1[31:0]	RW	64	Debug Breakpoint Value Register 5
0x454	DBGBVR5_EL1[63:32]	RW	64	Debug Breakpoint Value Register 5
0x458	DBGBCR5_EL1	RW	32	<a href="#">5.1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
0x45C–0x7FC	-	-	-	Reserved
0x800	DBGWVR0_EL1[31:0]	RW	64	Debug Watchpoint Value Register 0
0x804	DBGWVR0_EL1[63:32]			
0x808	DBGWCR0_EL1	RW	32	<a href="#">5.1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</a> on page 269
0x80C	-	-	-	Reserved
0x810	DBGWVR1_EL1[31:0]	RW	64	Debug Watchpoint Value Register 1
0x814	DBGWVR1_EL1[63:32]			
0x818	DBGWCR1_EL1	RW	32	<a href="#">5.1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</a> on page 269
0x81C	-	-	-	Reserved

Offset	Name	Type	Width	Description
0x820	DBGWVR2_EL1[31:0]	RW	64	Debug Watchpoint Value Register 2
0x824	DBGWVR2_EL1[63:32]			
0x828	DBGWCR2_EL1	RW	32	<a href="#">5.1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</a> on page 269
0x82C	-	-	-	Reserved
0x830	DBGWVR3_EL1[31:0]	RW	64	Debug Watchpoint Value Register 0,
0x834	DBGWVR3_EL1[63:32]			
0x838	DBGWCR3_EL1	RW	32	<a href="#">5.1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</a> on page 269
0x83C–0xCFC	-	-	-	Reserved
0xD00	MIDR	RO	32	<a href="#">3.1.64 MIDR_EL1, Main ID Register, EL1</a> on page 168
0xD04–0xD1C	-	-	-	Reserved
0xD20	EDPFR[31:0]	RO	64	<a href="#">3.1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1</a> on page 160
0xD24	EDPFR[63:32]			
0xD28	EDDFR[31:0]	RO	64	<a href="#">3.1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1</a> on page 160
0xD2C	EDDFR[63:32]			
0xD60–0xEFC	-	-	-	Reserved
0xF00	-	-	-	Reserved
0xF04–0xF9C	-	-	-	Reserved
0xFA0	DBGCLAIMSET_EL1	RW	32	<a href="#">5.1.3 DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1</a> on page 269
0xFA4	DBGCLAIMCLR_EL1	RW	32	Debug Claim Tag Clear Register
0xFA8	EDDEVAFF0	RO	32	External Debug Device Affinity Register 0
0xFAC	EDDEVAFF1	RO	32	External Debug Device Affinity Register 1
0xFB0	-	-	-	Reserved
0xFB4	-	-	-	Reserved
0xFB8	DBGAUTHSTATUS_EL1	RO	32	Debug Authentication Status Register
0xFBC	EDDEVARCH	RO	32	External Debug Device Architecture Register
0xFC0	EDDEVID2	RO	32	External Debug Device ID Register 2, <i>res0</i>
0xFC4	EDDEVID1	RO	32	<a href="#">5.2.7 EDDEVID1, External Debug Device ID Register 1</a> on page 281
0xFC8	EDDEVID	RO	32	<a href="#">5.2.6 EDDEVID, External Debug Device ID Register 0</a> on page 281
0xFCC	EDDEVTYPE	RO	32	External Debug Device Type Register
0xFD0	EDPIDR4	RO	32	<a href="#">5.2.15 EDPIDR4, External Debug Peripheral Identification Register 4</a> on page 289
0xFD4–0xFDC	EDPIDR5-7	RO	32	<a href="#">5.2.16 EDPIDRn, External Debug Peripheral Identification Registers 5-7</a> on page 290
0xFE0	EDPIDR0	RO	32	<a href="#">5.2.11 EDPIDR0, External Debug Peripheral Identification Register 0</a> on page 286
0xFE4	EDPIDR1	RO	32	<a href="#">5.2.12 EDPIDR1, External Debug Peripheral Identification Register 1</a> on page 286
0xFE8	EDPIDR2	RO	32	<a href="#">5.2.13 EDPIDR2, External Debug Peripheral Identification Register 2</a> on page 287
0xFEC	EDPIDR3	RO	32	<a href="#">5.2.14 EDPIDR3, External Debug Peripheral Identification Register 3</a> on page 288

Offset	Name	Type	Width	Description
0xFF0	EDCIDR0	RO	32	<a href="#">5.2.2 EDCIDR0, External Debug Component Identification Register 0 on page 278</a>
0xFF4	EDCIDR1	RO	32	<a href="#">5.2.3 EDCIDR1, External Debug Component Identification Register 1 on page 278</a>
0xFF8	EDCIDR2	RO	32	<a href="#">5.2.4 EDCIDR2, External Debug Component Identification Register 2 on page 279</a>
0xFFC	EDCIDR3	RO	32	<a href="#">5.2.5 EDCIDR3, External Debug Component Identification Register 3 on page 280</a>

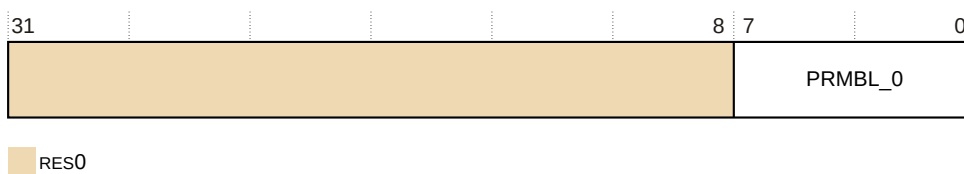
## 5.2.2 EDCIDR0, External Debug Component Identification Register 0

The EDCIDR0 provides information to identify an external debug component.

### Bit field descriptions

The EDCIDR0 is a 32-bit register.

**Figure 5-5: EDCIDR0 bit assignments**



#### RES0, [31:8]

*res0*

Reserved.

#### PRMBL\_0, [7:0]

**0x0D**

Preamble byte 0.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The EDCIDR0 can be accessed through the external debug interface, offset 0xFF0.

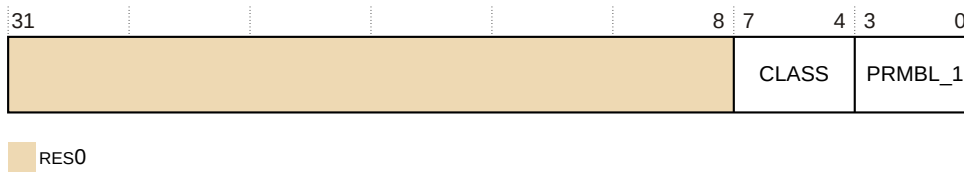
### 5.2.3 EDCIDR1, External Debug Component Identification Register 1

The EDCIDR1 provides information to identify an external debug component.

#### Bit field descriptions

The EDCIDR1 is a 32-bit register.

**Figure 5-6: EDCIDR1 bit assignments**



#### RES0, [31:8]

*res0*

Reserved.

#### CLASS, [7:4]

**0x9**

Debug component.

#### PRMBL\_1, [3:0]

**0x0**

Preamble.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

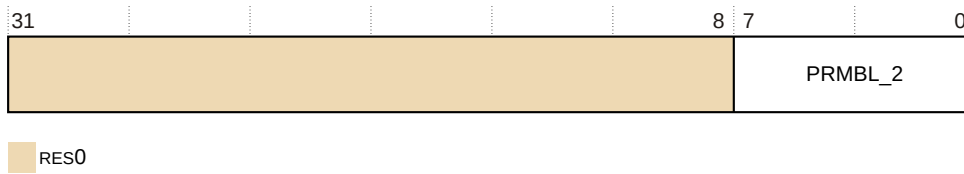
The EDCIDR1 can be accessed through the external debug interface, offset 0xFF4.

### 5.2.4 EDCIDR2, External Debug Component Identification Register 2

The EDCIDR2 provides information to identify an external debug component.

#### Bit field descriptions

The EDCIDR2 is a 32-bit register.

**Figure 5-7: EDCIDR2 bit assignments****RES0, [31:8]***res0*

Reserved.

**PRMBL\_2, [7:0]****0x05**

Preamble byte 2.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

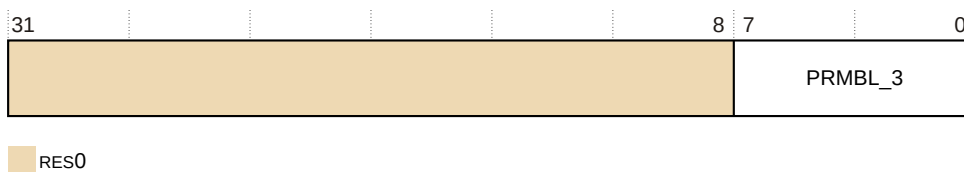
The EDCIDR2 can be accessed through the external debug interface, offset 0xFF8.

**5.2.5 EDCIDR3, External Debug Component Identification Register 3**

The EDCIDR3 provides information to identify an external debug component.

**Bit field descriptions**

The EDCIDR3 is a 32-bit register.

**Figure 5-8: EDCIDR3 bit assignments****RES0, [31:8]***res0*

Reserved.



**PRMBL\_3, [7:0]****0xB1**

Preamble byte 3.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

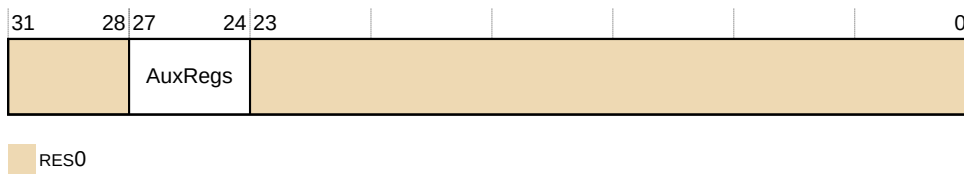
The EDCIDR3 can be accessed through the external debug interface, offset 0xFFC.

**5.2.6 EDDEVID, External Debug Device ID Register 0**

The EDDEVID provides extra information for external debuggers about features of the debug implementation.

**Bit field descriptions**

The EDDEVID is a 32-bit register.

**Figure 5-9: EDDEVID bit assignments****RES0, [31:28]****res0**

Reserved.

**AuxRegs, [27:24]**

Indicates support for Auxiliary registers:

**0x0**

None supported.

**RES0, [23:0]****res0**

Reserved.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The EDDEVID can be accessed through the external debug interface, offset 0xFC8.

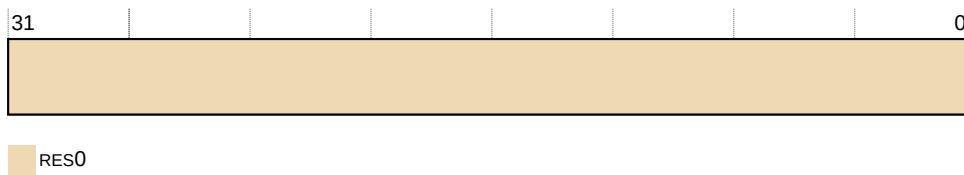
## 5.2.7 EDDEVID1, External Debug Device ID Register 1

The EDDEVID1 provides extra information for external debuggers about features of the debug implementation.

### Bit field descriptions

The EDDEVID1 is a 32-bit register.

**Figure 5-10: EDDEVID1 bit assignments**



### RES0, [31:0]

*res0*

Reserved.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The EDDEVID1 can be accessed through the external debug interface, offset 0xFC4.

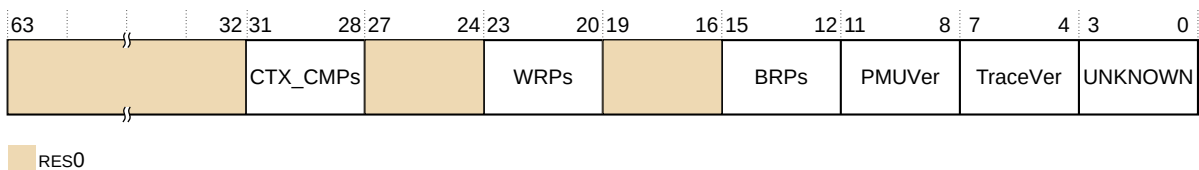
## 5.2.8 EDDFR, External Debug Feature Register

The EDDFR provides top level information about the debug system in AArch64.

### Bit field descriptions

The EDDFR is a 64-bit register.

**Figure 5-11: EDDFR bit assignments**



### RES0, [63:32]

RES0

Reserved.

**CTX\_CMPs, [31:28]**

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

**RES0, [27:24]**

**RES0** Reserved.

**WRPs, [23:20]**

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

**RES0, [19:16]**

**RES0** Reserved.

**BRPs, [15:12]**

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

**PMUVer, [11:8]**

Performance Monitors extension version. Indicates whether system register interface to Performance Monitors extension is implemented. Defined values are:

0x0000	Performance Monitors extension system registers not implemented.
0x0001	Performance Monitors extension system registers implemented, PMUv3.
0x1111	<b>IMPLEMENTATION DEFINED</b> form of performance monitors supported, PMUv3 not supported.

All other values are reserved.

**TraceVer [7:4]**

Trace support. Indicates whether system register interface to a trace macrocell is implemented. Defined values are:

0x0000	Trace macrocell system registers not implemented.
0x0001	Trace macrocell system registers implemented.

All other values are reserved.

A value of 0x0000 only indicates that no system register interface to a trace macrocell is implemented. A trace macrocell might nevertheless be implemented without a system register interface.

**UNKNOWN, [3:0]**

**UNKNOWN** Reserved.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

EDDFR[31:0] can be accessed through the external debug interface, offset 0xD28.

EDDFR[63:32] can be accessed through the external debug interface, offset 0xD2C.

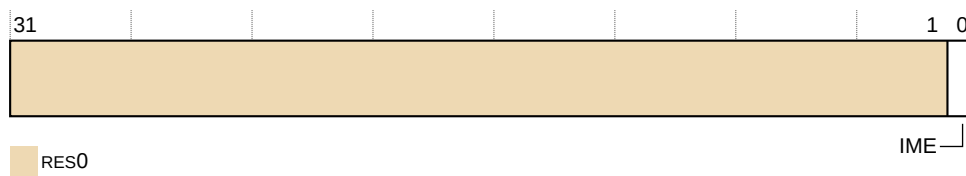
### 5.2.9 EDITCTRL, External Debug Integration Mode Control Register

The EDITCTRL enables the external debug to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the core, for integration testing or topology detection.

#### Bit field descriptions

The EDITCTRL is a 32-bit register.

**Figure 5-12: EDITCTRL bit assignments**



#### [31:1]

**RES0** Reserved.

#### IME, [0]

Integration Mode Enable.

**RES0.** The device does not revert to an integration mode to enable integration testing or topology detection.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

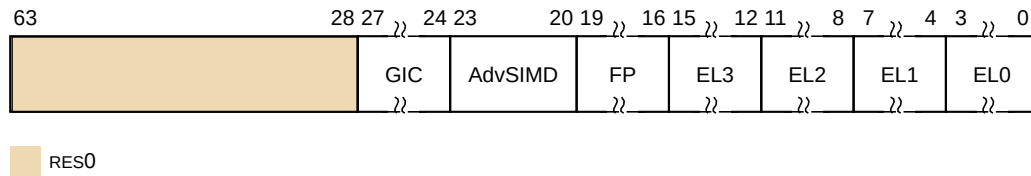
The EDITCTRL can be accessed through the external debug interface, offset 0xF00.

### 5.2.10 EDPFR, External Debug Processor Feature Register

The EDPFR provides additional information about implemented PE features in AArch64.

#### Bit field descriptions

The EDPFR is a 64-bit register.

**Figure 5-13: EDPFR bit assignments****RES0, [63:28]**

RES0      Reserved.

**GIC, [27:24]**

System register GIC interface. Defined values are:

0x0	No System register interface to the GIC is supported.
0x1	System register interface to the GIC CPU interface is supported.

All other values are reserved.

**AdvSIMD, [23:20]**

Advanced SIMD. Defined values are:

0x0	Advanced SIMD is implemented.
0xF	Advanced SIMD is not implemented.

All other values are reserved.

**FP, [19:16]**

Floating-point. Defined values are:

0x0	Floating-point is implemented.
0xF	Floating-point is not implemented.

All other values are reserved.

**EL3 handling, [15:12]**

EL3 exception handling:

0x2	Instructions can be executed at EL3.
-----	--------------------------------------

**EL2 handling, [11:8]**

EL2 exception handling:

0x2	Instructions can be executed at EL2.
-----	--------------------------------------

**EL1 handling, [7:4]**

EL1 exception handling. The possible values are:

0x2            Instructions can be executed at EL1.

**ELO handling, [3:0]**

ELO exception handling. The possible values are:

0x2            Instructions can be executed at ELO.

Bit fields and details not provided in this description are architecturally defined. See the Arm® *Architecture Reference Manual Armv8, for A-profile architecture*.

The EDPFR[31:0] can be accessed through the external debug interface, offset 0xD20.

The EDPFR[63:32] can be accessed through the external debug interface, offset 0xD24.

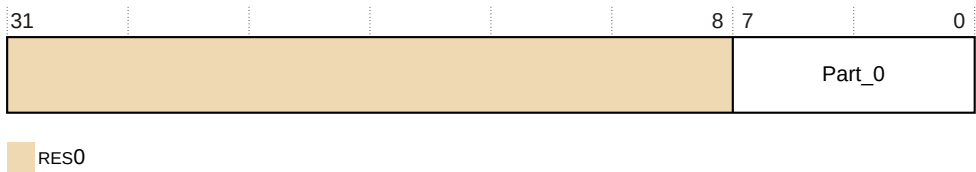
**5.2.11 EDPIDR0, External Debug Peripheral Identification Register 0**

The EDPIDR0 provides information to identify an external debug component.

**Bit field descriptions**

The EDPIDR0 is a 32-bit register.

**Figure 5-14: EDPIDR0 bit assignments**



**RES0, [31:8]**

*res0*

Reserved.

**Part\_0, [7:0]**

0x43            Least significant byte of the debug part number.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® *Architecture Reference Manual Armv8, for A-profile architecture*.

The EDPIDR0 can be accessed through the external debug interface, offset 0xFE0.

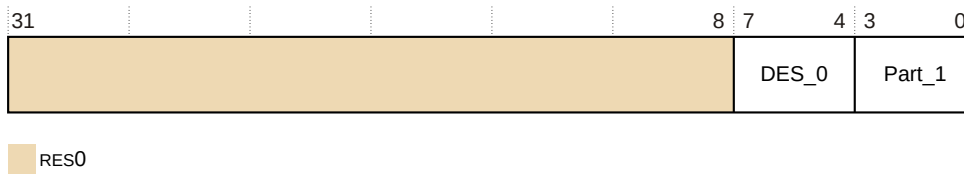
## 5.2.12 EDPIDR1, External Debug Peripheral Identification Register 1

The EDPIDR1 provides information to identify an external debug component.

### Bit field descriptions

The EDPIDR1 is a 32-bit register.

**Figure 5-15: EDPIDR1 bit assignments**



#### RES0, [31:8]

**res0**

Reserved.

#### DES\_0, [7:4]

**0xB**

Arm Limited. This is the least significant nibble of JEP106 ID code.

#### Part\_1, [3:0]

**0xD**

Most significant nibble of the debug part number.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

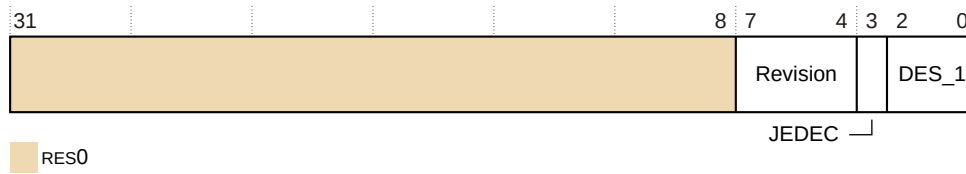
The EDPIDR1 can be accessed through the external debug interface, offset 0xFE4.

## 5.2.13 EDPIDR2, External Debug Peripheral Identification Register 2

The EDPIDR2 provides information to identify an external debug component.

### Bit field descriptions

The EDPIDR2 is a 32-bit register.

**Figure 5-16: EDPIDR2 bit assignments****RES0, [31:8]**

*res0* Reserved.

**Revision, [7:4]**

0x2 r1p1.

**JEDEC, [3]**

0b1 RAO. Indicates a JEP106 identity code is used.

**DES\_1, [2:0]**

0b011 Arm Limited. This is the most significant nibble of JEP106 ID code.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

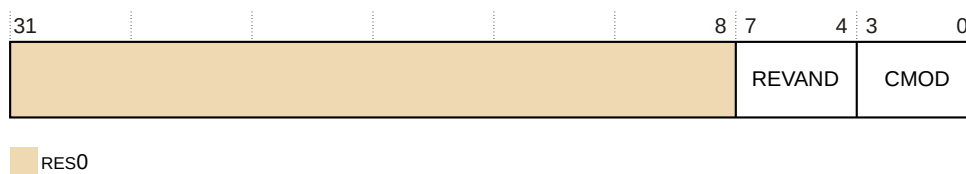
The EDPIDR2 can be accessed through the external debug interface, offset 0xFE8.

**5.2.14 EDPIDR3, External Debug Peripheral Identification Register 3**

The EDPIDR3 provides information to identify an external debug component.

**Bit field descriptions**

The EDPIDR3 is a 32-bit register.

**Figure 5-17: EDPIDR3 bit assignments**



**RES0, [31:8]**  
*res0*  
Reserved.

**REVAND, [7:4]**  
**0x0**  
Part minor revision.

**CMOD, [3:0]**  
**0x0**  
Customer modified.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The EDPIDR3 can be accessed through the external debug interface, offset 0xFEC.

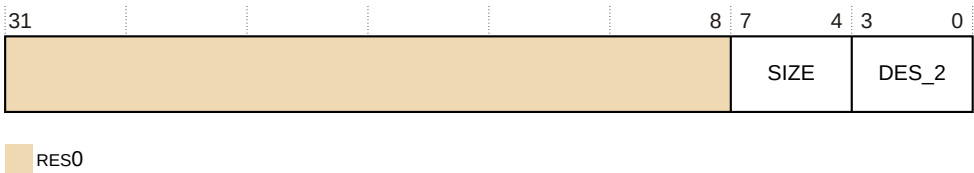
5.2.15 EDPIDR4, External Debug Peripheral Identification Register 4

The EDPIDR4 provides information to identify an external debug component.

Bit field descriptions

The EDPIDR4 is a 32-bit register.

Figure 5-18: EDPIDR4 bit assignments



**RES0, [31:8]**  
*res0*  
Reserved.

**SIZE, [7:4]**  
**0x0**  
Size of the component. Log<sub>2</sub> the number of 4KB pages from the start of the component to the end of the component ID registers.

**DES\_2, [3:0]****0x4**

Arm Limited This is the least significant nibble JEP106 continuation code.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The EDPIDR4 can be accessed through the external debug interface, offset 0xFD0.

**5.2.16 EDPIDRn, External Debug Peripheral Identification Registers 5-7**

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers.

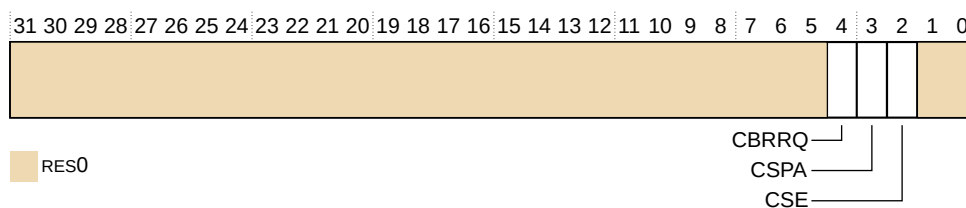
They are reserved for future use and are **RES0**.

**5.2.17 EDRCR, External Debug Reserve Control Register**

The EDRCR is part of the Debug registers functional group. This register is used to allow imprecise entry to Debug state and clear sticky bits in EDSCR.

**Bit field descriptions**

The EDRCR is a 32-bit register.

**Figure 5-19: EDRCR bit assignments****RES0, [31:5]**

**RES0** Reserved.

**CBRRQ, [4]**

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. |

**CSPA, [3]**

Clear Sticky Pipeline Advance. This bit is used to clear the EDSCR.PipeAdv bit to 0. The actions on writing to this bit are:

0	No action.
1	Clear the EDSCR.PipeAdv bit to 0.

**CSE, [2]**

Clear Sticky Error. Used to clear the EDSCR cumulative error bits to 0. The actions on writing to this bit are:

0	No action
1	Clear the EDSCR.{TXU, RXO, ERR} bits, and, if the core is in Debug state, the EDSCR.ITO bit, to 0.

**RES0, [1:0]**

<b>RES0</b>	Reserved.
-------------	-----------

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The EDRCR can be accessed through the external debug interface, offset 0x090.

## 5.3 AArch64 PMU registers

This chapter describes the AArch64 PMU registers and shows examples of how to use them.

### 5.3.1 AArch64 PMU register summary

The PMU counters and their associated control registers are accessible in the AArch64 Execution state with `MRS` and `MSR` instructions.

The following table gives a summary of the Cortex®-A65AE PMU registers in the AArch64 Execution state. For those registers that are not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**Table 5-3: PMU register summary in the AArch64 Execution state**

Name	Type	Width	Reset	Description
PMCR_ELO	RW	32	0x41473040	5.3.4 PMCR_ELO, Performance Monitors Control Register, ELO on page 298
PMCNTENSET_ELO	RW	32	UNK	Performance Monitors Count Enable Set Register

Name	Type	Width	Reset	Description
PMCNTENCLR_ELO	RW	32	UNK	Performance Monitors Count Enable Clear Register
PMOVSCLR_ELO	RW	32	UNK	Performance Monitors Overflow Flag Status Register
PMSWINC_ELO	WO	32	UNK	Performance Monitors Software Increment Register
PMSELR_ELO	RW	32	UNK	Performance Monitors Event Counter Selection Register
PMCEID0_ELO	RO	64	<ul style="list-style-type: none"> <li>If L2 is present: 0x000000007FFF7FFF</li> <li>If L2 is not present, but L3 is present: 0x000000007EFF7FFF</li> <li>If neither L2 nor L3 are present: 0x000000007E3F7FFF</li> </ul>	<a href="#">5.3.2 PMCEID0_ELO, Performance Monitors Common Event Identification Register 0, ELO</a> on page 293
PMCEID1_ELO	RO	64	<ul style="list-style-type: none"> <li>If both L2 and L3 are present: 0x0000000001F0AE7F</li> <li>If L2 or L3 is present, but not both: 0x0000000001F0A07F</li> <li>If neither L2 nor L3 are present: 0x0000000001F0A07E</li> </ul>	<a href="#">5.3.3 PMCEID1_ELO, Performance Monitors Common Event Identification Register 1, ELO</a> on page 296
PMCCNTR_ELO	RW	64	UNK	Performance Monitors Cycle Count Register
PMXEVTYPER_ELO	RW	32	UNK	Performance Monitors Selected Event Type and Filter Register
PMCCFILTR_ELO	RW	32	UNK	Performance Monitors Cycle Count Filter Register
PMXVCNTR_ELO	RW	32	UNK	Performance Monitors Selected Event Count Register
PMUSERENR_ELO	RW	32	UNK	Performance Monitors User Enable Register
PMINTENSET_EL1	RW	32	UNK	Performance Monitors Interrupt Enable Set Register
PMINTENCLR_EL1	RW	32	UNK	Performance Monitors Interrupt Enable Clear Register

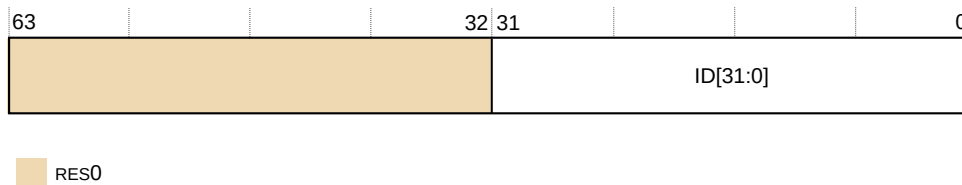
Name	Type	Width	Reset	Description
PMOVSET_ELO	RW	32	UNK	Performance Monitors Overflow Flag Status Set Register
PMEVCNTR0_ELO	RW	32	UNK	Performance Monitors Event Count Registers
PMEVCNTR1_ELO	RW	32	UNK	
PMEVCNTR2_ELO	RW	32	UNK	
PMEVCNTR3_ELO	RW	32	UNK	
PMEVCNTR4_ELO	RW	32	UNK	
PMEVCNTR5_ELO	RW	32	UNK	
PMEVTYPER0_ELO	RW	32	UNK	Performance Monitors Event Type Registers
PMEVTYPER1_ELO	RW	32	UNK	
PMEVTYPER2_ELO	RW	32	UNK	
PMEVTYPER3_ELO	RW	32	UNK	
PMEVTYPER4_ELO	RW	32	UNK	
PMEVTYPER5_ELO	RW	32	UNK	
PMCCFILTR_ELO	RW	32	UNK	Performance Monitors Cycle Count Filter Register

### 5.3.2 PMCEID0\_ELO, Performance Monitors Common Event Identification Register 0, ELO

The PMCEID0\_ELO defines which common architectural and common microarchitectural feature events are implemented.

#### Bit field descriptions

**Figure 5-20: PMCEID0\_ELO bit assignments**



#### ID[31:0], [31:0]

Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

The following table shows the PMCEID0\_ELO bit assignments with event implemented or not implemented when the associated bit is set to 1 or 0. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information about these events.

**Table 5-4: PMU events**

Bit	Event number	Event mnemonic	Description
[31]	0x1F	L1D_CACHE_ALLOCATE	L1 Data cache allocate:  0 This event is not implemented.
[30]	0x1E	CHAIN	Chain. For odd-numbered counters, counts once for each overflow of the preceding even-numbered counter. For even-numbered counters, does not count:  1 This event is implemented.
[29]	0x1D	BUS_CYCLES	Bus cycle:  1 This event is implemented.
[28]	0x1C	TTBR_WRITE_RETIRED	TTBR write, architecturally executed, condition check pass - write to translation table base:  1 This event is implemented.
[27]	0x1B	INST_SPEC	Instruction speculatively executed:  1 This event is implemented.
[26]	0x1A	MEMORY_ERROR	Local memory error:  1 This event is implemented.
[25]	0x19	BUS_ACCESS	Bus access:  1 This event is implemented.
[24]	0x18	L2D_CACHE_WB	L2 unified cache Write-Back:  0 This event is not implemented if the Cortex®-A65AE core has been configured without an L2 cache. 1 This event is implemented if the Cortex®-A65AE core has been configured with an L2 cache.
[23]	0x17	L2D_CACHE_REFILL	L2 unified cache refill:  0 This event is not implemented if the Cortex®-A65AE core has been configured without an L2 and L3 cache. If configured with only an L3 cache, the L3 event will become an L2 event. 1 This event is implemented if the Cortex®-A65AE core has been configured with an L2 or L3 cache.
[22]	0x16	L2D_CACHE	L2 unified cache access:  0 This event is not implemented if the Cortex®-A65AE core has been configured without an L2 and L3 cache. If configured with only an L3 cache, the L3 event will become an L2 event. 1 This event is implemented if the Cortex®-A65AE core has been configured with an L2 or L3 cache.
[21]	0x15	L1D_CACHE_WB	L1 Data cache Write-Back:  1 This event is implemented.
[20]	0x14	L1I_CACHE	L1 Instruction cache access:  1 This event is implemented.

Bit	Event number	Event mnemonic	Description
[19]	0x13	MEM_ACCESS	Data memory access: 1 This event is implemented.
[18]	0x12	BR_PRED	Predictable branch speculatively executed: 1 This event is implemented.
[17]	0x11	CPU_CYCLES	Cycle: 1 This event is implemented.
[16]	0x10	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed: 1 This event is implemented.
[15]	0x0F	UNALIGNED_LDST_RETIRED	Instruction architecturally executed, condition check pass - unaligned load or store: 0 This event is not implemented.
[14]	0x0E	BR_RETURN_RETIRED	Instruction architecturally executed, condition check pass - procedure return: 1 This event is implemented.
[13]	0x0D	BR_IMMED_RETIRED	Instruction architecturally executed - immediate branch: 1 This event is implemented.
[12]	0x0C	PC_WRITE_RETIRED	Instruction architecturally executed, condition check pass - software change of the PC: 1 This event is implemented.
[11]	0x0B	CID_WRITE_RETIRED	Instruction architecturally executed, condition check pass - write to CONTEXTIDR: 1 This event is implemented.
[10]	0x0A	EXC_RETURN	Instruction architecturally executed, condition check pass - exception return: 1 This event is implemented.
[9]	0x09	EXC_TAKEN	Exception taken: 1 This event is implemented.
[8]	0x08	INST_RETIRED	Instruction architecturally executed: 1 This event is implemented.
[7]	0x07	ST_RETIRED	Instruction architecturally executed, condition check pass - store: 1 This event is implemented.
[6]	0x06	LD_RETIRED	Instruction architecturally executed, condition check pass - load: 1 This event is implemented.
[5]	0x05	L1D_TLB_REFILL	L1 Data TLB refill: 1 This event is implemented.
[4]	0x04	L1D_CACHE	L1 Data cache access: 1 This event is implemented.

Bit	Event number	Event mnemonic	Description
[3]	0x03	L1D_CACHE_REFILL	L1 Data cache refill: 1 This event is implemented.
[2]	0x02	L1I_TLB_REFILL	L1 Instruction TLB refill: 1 This event is implemented.
[1]	0x01	L1I_CACHE_REFILL	L1 Instruction cache refill: 1 This event is implemented.
[0]	0x00	SW_INCR	Instruction architecturally executed, condition check pass - software increment: 1 This event is implemented.

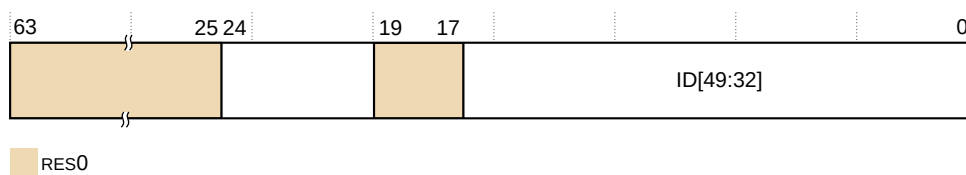
Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 5.3.3 PMCEID1\_ELO, Performance Monitors Common Event Identification Register 1, ELO

The PMCEID1\_ELO defines which common architectural and common microarchitectural feature events are implemented.

#### Bit field descriptions

**Figure 5-21: PMCEID1\_ELO bit assignments**



#### ID[49:32], [17:0]

Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

For each bit described in the following table, the event is implemented if the bit is set to 1, or not implemented if the bit is set to 0.

**Table 5-5: PMU common events**

Bit	Event number	Event mnemonic	Description
[24]	0x38	REMOTE_ACCESS_RD	Access to another socket in a multi-socket system. 1 This event is implemented.



Bit	Event number	Event mnemonic	Description
[23]	0x37	LL_CACHE_MISS_RD	Last Level cache miss, read. 1 This event is implemented.
[22]	0x36	LL_CACHE_RD	Last Level cache access, read. 1 This event is implemented.
[21]	0x35	ITLB_WALK	Access to instruction TLB that caused a page table walk. 1 This event is implemented.
[20]	0x34	DTLB_WALK	Access to data TLB that caused a page table walk. 1 This event is implemented.
[16]	0x30	L2I_TLB	Attributable Level 2 instruction TLB access. 0 This event is not implemented.
[15]	0x2F	L2D_TLB	Attributable Level 2 data or unified TLB access. 1 This event is implemented.
[14]	0x2E	L2I_TLB_REFILL	Attributable Level 2 instruction TLB refill. 0 This event is not implemented.
[13]	0x2D	L2D_TLB_REFILL	Attributable Level 2 data or unified TLB refill. 1 This event is implemented.
[12]	0x2C	L3D_CACHE_WB	Attributable Level 3 unified cache write-back. 0 This event is not implemented.
[11]	0x2B	L3D_CACHE	Attributable Level 3 unified cache access. 1 This event is implemented if L2 and L3 are present. 0 This event is not implemented if L2 and L3 are not present.
[10]	0x2A	L3D_CACHE_REFILL	Attributable Level 3 unified cache refill. 1 This event is implemented if L2 and L3 are present. 0 This event is not implemented if L2 and L3 are not present.
[9]	0x29	L3D_CACHE_ALLOCATE	Attributable Level 3 unified cache allocation without refill. 1 This event is implemented if L2 and L3 are present. 0 This event is not implemented if L2 and L3 are not present.
[8]	0x28	L2I_CACHE_REFILL	Attributable Level 2 instruction cache refill. 0 This event is not implemented.
[7]	0x27	L2I_CACHE	Attributable Level 2 instruction cache access. 0 This event is not implemented.
[6]	0x26	L1I_TLB	Level 1 instruction TLB access. 1 This event is implemented.

Bit	Event number	Event mnemonic	Description
[5]	0x25	L1D_TLB	Level 1 data TLB access. 1 This event is implemented.
[4]	0x24	STALL_BACKEND	No operation issued due to backend. 1 This event is implemented.
[3]	0x23	STALL_FRONTEND	No operation issued due to the frontend. 1 This event is implemented.
[2]	0x22	BR_MIS_PRED_RETIRED	Instruction architecturally executed, mispredicted branch. 1 This event is implemented.
[1]	0x21	BR_RETIRED	Instruction architecturally executed, branch. 1 This event is implemented.
[0]	0x20	L2D_CACHE_ALLOCATE	Level 2 data cache allocation without refill. 1 This event is implemented.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### 5.3.4 PMCR\_ELO, Performance Monitors Control Register, ELO

The PMCR\_ELO provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

#### Bit field descriptions

**Figure 5-22: PMCR\_ELO bit assignments**

31		24	23		16	15		11	10		7	6	5	4	3	2	1	0
IMP				IDCODE				N				L	C	D	P	X	E	

RES0

#### IMP, [31:24]

Implementer code:

0x41 Arm.

This is a read-only field.

#### IDCODE, [23:16]

Identification code:

0x47 Cortex®-A65AE.

This is a read-only field.

### **N, [15:11]**

Number of event counters.

0b00110 Six counters.

### **RES0, [10:7]**

**RES0** Reserved.

### **LC, [6]**

Long cycle count enable. Determines which PMCCNTR\_ELO bit generates an overflow recorded in PMOVSR[31]. The possible values are:

0	Overflow on increment that changes PMCCNTR_ELO[31] from 1 to 0.
1	Overflow on increment that changes PMCCNTR_ELO[63] from 1 to 0.

### **DP, [5]**

Disable cycle counter, PMCCNTR\_ELO when event counting is prohibited:

0	Cycle counter operates regardless of the non-invasive debug authentication settings. This is the reset value.
1	Cycle counter is disabled if non-invasive debug is not permitted and enabled.

This bit is read/write.

### **X, [4]**

Export enable. This bit permits events to be exported to another debug device, such as a trace macrocell, over an event bus:

0	Export of events is disabled. This is the reset value.
1	Export of events is enabled.

This bit is read/write and does not affect the generation of Performance Monitors interrupts on the **nPMUIRQ** pin.

### **D, [3]**

Clock divider:

0	When enabled, PMCCNTR_ELO counts every clock cycle. This is the reset value.
1	When enabled, PMCCNTR_ELO counts every 64 clock cycles.

This bit is read/write.

### **C, [2]**

Clock counter reset. This bit is WO. The effects of writing to this bit are:

0	No action. This is the reset value.
1	Reset PMCCNTR_ELO to 0.

This bit is always RAZ.

Resetting PMCCNTR\_ELO does not clear the PMCCNTR\_ELO overflow bit to 0. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

## P, [1]

Event counter reset. This bit is WO. The effects of writing to this bit are:

0	No action. This is the reset value.
1	Reset all event counters, not including PMCCNTR_ELO, to zero.

This bit is always RAZ.

In Non-secure ELO and EL1, a write of 1 to this bit does not reset event counters that MDCR\_EL2.HPMN reserves for EL2 use.

In EL2 and EL3, a write of 1 to this bit resets all the event counters.

Resetting the event counters does not clear any overflow bits to 0.

## E, [0]

Enable. The possible values of this bit are:

0	All counters, including PMCCNTR_ELO, are disabled. This is the reset value.
1	All counters are enabled.

This bit is RW.

In Non-secure ELO and EL1, this bit does not affect the operation of event counters that MDCR\_EL2.HPMN reserves for EL2 use.

On Warm reset, the field resets to 0.

## Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 5.4 Memory-mapped PMU registers

This chapter describes the memory-mapped PMU registers and shows examples of how to use them.

### 5.4.1 Memory-mapped PMU register summary

There are PMU registers that are accessible through the external debug interface.

These registers are listed in the following table. For those registers that are not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

**Table 5-6: Memory-mapped PMU register summary**

Offset	Name	Type	Description
0x000	PMEVCNTR0_ELO	RW	Performance Monitor Event Count Register 0
0x004	-	-	Reserved
0x008	PMEVCNTR1_ELO	RW	Performance Monitor Event Count Register 1
0x00C	-	-	Reserved
0x010	PMEVCNTR2_ELO	RW	Performance Monitor Event Count Register 2
0x014	-	-	Reserved
0x018	PMEVCNTR3_ELO	RW	Performance Monitor Event Count Register 3
0x01C	-	-	Reserved
0x020	PMEVCNTR4_ELO	RW	Performance Monitor Event Count Register 4
0x024	-	-	Reserved
0x028	PMEVCNTR5_ELO	RW	Performance Monitor Event Count Register 5
0x02C–0x0F4	-	-	Reserved
0x0F8	PMCCNTR_ELO[31:0]	RW	Performance Monitor Cycle Count Register
0x0FC	PMCCNTR_ELO[63:32]	RW	
0x200	PMPCSR[31:0]	RO	Program Counter Sample Register
0x204	PMPCSR[63:32]		
0x208	PMCID1SR	RO	CONTEXTIDR_EL1 Sample Register
0x20C	PMVIDSR	RO	VMID Sample Register
0x220	PMPCSR[31:0]	RO	Program Counter Sample Register (alias)
0x224	PMPCSR[63:32]		
0x228	PMCID1SR	RO	CONTEXTIDR_EL1 Sample Register (alias)
0x22C	PMCID2SR	RO	CONTEXTIDR_EL2 Sample Register
0x230–0x3FC	-	-	Reserved

Offset	Name	Type	Description
0x418–0x478	-	-	Reserved
0x47C	PMCCFILTR_ELO	RW	Performance Monitor Cycle Count Filter Register
0x600	PMPCSSR_LO	RO	5.5.2 PMPCSSR, PMU Snapshot Program Counter Sample Register on page 312
0x604	PMPCSSR_HI	RO	
0x608	PMCIDSSR	RO	5.5.3 PMCIDSSR, PMU Snapshot CONTEXTIDR_EL1 Sample Register on page 313
0x60C	PMCID2SSR	RO	5.5.4 PMCID2SSR, PMU Snapshot CONTEXTIDR_EL2 Sample Register on page 314
0x610	PMSSSR	RO	5.5.5 PMSSSR, PMU Snapshot Status Register on page 314
0x614	PMOVSSR	RO	5.5.6 PMOVSSR, PMU Snapshot Overflow Status Register on page 315
0x618	PMCCNTSR_LO	RO	5.5.7 PMCCNTSR, PMU Snapshot Cycle Counter Register on page 315
0x61C	PMCCNTSR_HI	RO	
0x620+ 4×n	PMEVCNTSRn	RO	5.5.8 PMEVCNTSRn, PMU Snapshot Cycle Counter Registers 0-5 on page 316
0x6F0	PMSSCR	WO	5.5.9 PMSSCR, PMU Snapshot Capture Register on page 316
0xC00	PMCNTENSET_ELO	RW	Performance Monitor Count Enable Set Register
0xC04–0xC1C	-	-	Reserved
0xC20	PMCNTENCLR_ELO	RW	Performance Monitor Count Enable Clear Register
0xC24–0xC3C	-	-	Reserved
0xC40	PMINTENSET_EL1	RW	Performance Monitor Interrupt Enable Set Register
0xC44–0xC5C	-	-	Reserved
0xC60	PMINTENCLR_EL1	RW	Performance Monitor Interrupt Enable Clear Register
0xC64–0xC7C	-	-	Reserved
0xC80	PMOVSCLR_ELO	RW	Performance Monitor Overflow Flag Status Register
0xC84–0xC9C	-	-	Reserved
0xCA0	PMSWINC_ELO	WO	Performance Monitor Software Increment Register
0xCA4–0xCBC	-	-	Reserved
0xCC0	PMOVSSET_ELO	RW	Performance Monitor Overflow Flag Status Set Register
0xCC4–0xDFC	-	-	Reserved
0xE00	PMCFGR	RO	5.4.2 PMCFGR, Performance Monitors Configuration Register on page 304

Offset	Name	Type	Description
0xE04	PMCR_ELO	RW	Performance Monitors Control Register.  This register is distinct from the PMCR_ELO System register. It does not have the same value.
0xE08–0xE1C	-	-	Reserved
0xE20	PMCEID0	RO	<a href="#">5.3.2 PMCEID0_ELO, Performance Monitors Common Event Identification Register 0, ELO</a> on page 293
0xE24	PMCEID1	RO	<a href="#">5.3.3 PMCEID1_ELO, Performance Monitors Common Event Identification Register 1, ELO</a> on page 296
0xE28	PMCEID2	RO	Reserved
0xE2C	PMCEID3	RO	Reserved
0xFA4	-	-	Reserved
0xFA8	PMDEVAFF0	RO	<a href="#">3.1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1</a> on page 169
0xFAC	PMDEVAFF1	RO	<a href="#">3.1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1</a> on page 169
0xFB8	PMAUTHSTATUS	RO	Performance Monitor Authentication Status Register
0xFBC	PMDEVARCH	RO	Performance Monitor Device Architecture Register
0xFC0–0xFC8	-	-	Reserved
0xFCC	PMDEVTYPE	RO	Performance Monitor Device Type Register
0xFD0	PMPIDR4	RO	<a href="#">5.4.11 PMPIDR4, Performance Monitors Peripheral Identification Register 4</a> on page 311
0xFD4	PMPIDR5	RO	<a href="#">5.4.12 PMPIDRn, Performance Monitors Peripheral Identification Register 5-7</a> on page 312
0xFD8	PMPIDR6	RO	
0xFDC	PMPIDR7	RO	
0xFE0	PMPIDR0	RO	<a href="#">5.4.7 PMPIDR0, Performance Monitors Peripheral Identification Register 0</a> on page 308
0xFE4	PMPIDR1	RO	<a href="#">5.4.8 PMPIDR1, Performance Monitors Peripheral Identification Register 1</a> on page 308
0xFE8	PMPIDR2	RO	<a href="#">5.4.9 PMPIDR2, Performance Monitors Peripheral Identification Register 2</a> on page 309
0xFEC	PMPIDR3	RO	<a href="#">5.4.10 PMPIDR3, Performance Monitors Peripheral Identification Register 3</a> on page 310
0xFF0	PMCIDR0	RO	<a href="#">5.4.3 PMCIDR0, Performance Monitors Component Identification Register 0</a> on page 305

Offset	Name	Type	Description
0xFF4	PMCIDR1	RO	5.4.4 PMCIDR1, Performance Monitors Component Identification Register 1 on page 305
0xFF8	PMCIDR2	RO	5.4.5 PMCIDR2, Performance Monitors Component Identification Register 2 on page 306
0xFFC	PMCIDR3	RO	5.4.6 PMCIDR3, Performance Monitors Component Identification Register 3 on page 307

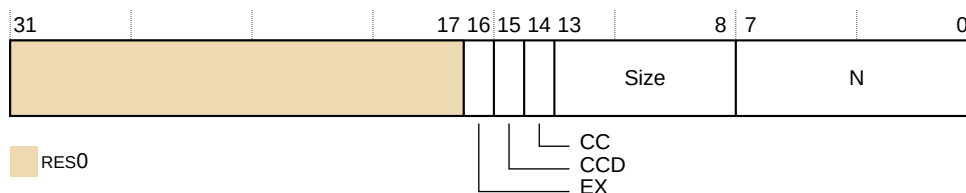
## 5.4.2 PMCFGR, Performance Monitors Configuration Register

The PMCFGR contains PMU specific configuration data.

### Bit field descriptions

The PMCFGR is a 32-bit register.

**Figure 5-23: PMCFGR bit assignments**



### RES0, [31:17]

*res0* Reserved.

### EX, [16]

Export supported. The value is:

1 Export is supported. PMCR\_ELO.EX is read/write.

### CCD, [15]

Cycle counter has pre-scale. The value is:

1 PMCR\_ELO.D is read/write.

### CC, [14]

Dedicated cycle counter supported. The value is:

1 Dedicated cycle counter is supported.



**Size, [13:8]**

Counter size. The value is:

0b111111 64-bit counters.

**N, [7:0]**

Number of event counters. The value is:

0x06 Six counters.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The PMCFGR can be accessed through the external debug interface, offset 0xE00.

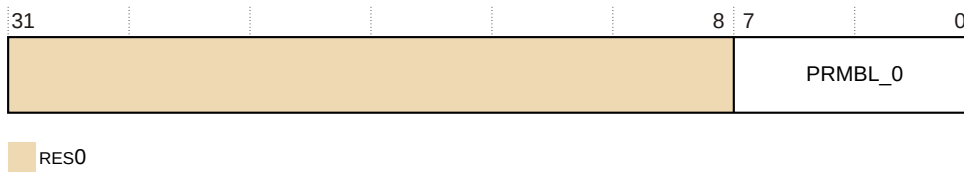
### 5.4.3 PMCIDR0, Performance Monitors Component Identification Register 0

The PMCIDR0 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMCIDR0 is a 32-bit register.

**Figure 5-24: PMCIDR0 bit assignments**

**RES0, [31:8]**

*res0*

Reserved.

**PRMBL\_0, [7:0]**

0x0D

Preamble byte 0.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The PMCIDR0 can be accessed through the external debug interface, offset 0xFF0.

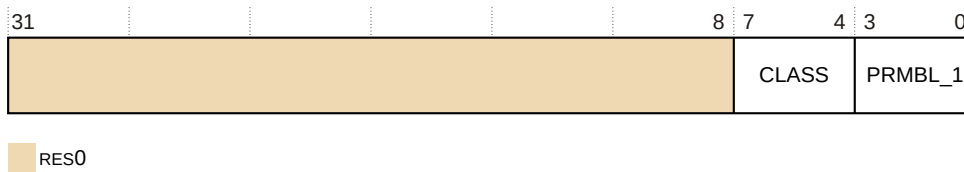
## 5.4.4 PMCIDR1, Performance Monitors Component Identification Register 1

The PMCIDR1 provides information to identify a Performance Monitor component.

### Bit field descriptions

The PMCIDR1 is a 32-bit register.

**Figure 5-25: PMCIDR1 bit assignments**



#### RES0, [31:8]

*res0*

Reserved.

#### CLASS, [7:4]

**0x9**

Debug component.

#### PRMBL\_1, [3:0]

**0x0**

Preamble byte 1.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

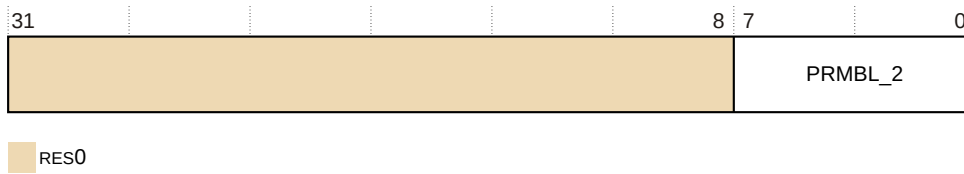
The PMCIDR1 can be accessed through the external debug interface, offset 0xFF4.

## 5.4.5 PMCIDR2, Performance Monitors Component Identification Register 2

The PMCIDR2 provides information to identify a Performance Monitor component.

### Bit field descriptions

The PMCIDR2 is a 32-bit register.

**Figure 5-26: PMCIDR2 bit assignments****RES0, [31:8]***res0*

Reserved.

**PRMBL\_2, [7:0]****0x05**

Preamble byte 2.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

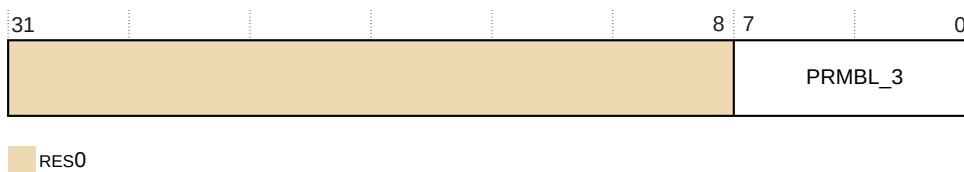
The PMCIDR2 can be accessed through the external debug interface, offset 0xFF8.

## 5.4.6 PMCIDR3, Performance Monitors Component Identification Register 3

The PMCIDR3 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMCIDR3 is a 32-bit register.

**Figure 5-27: PMCIDR3 bit assignments****RES0, [31:8]***res0*

Reserved.

**PRMBL\_3, [7:0]****0xB1**

Preamble byte 3.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

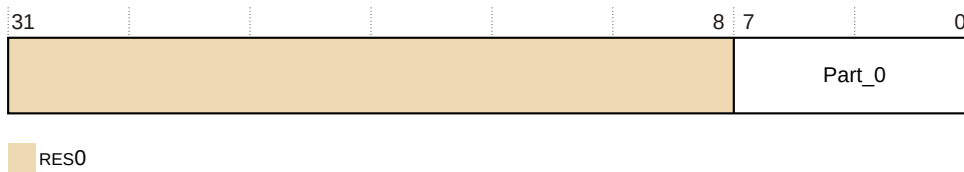
The PMCIDR3 can be accessed through the external debug interface, offset 0xFFC.

**5.4.7 PMPIDR0, Performance Monitors Peripheral Identification Register 0**

The PMPIDR0 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMPIDR0 is a 32-bit register.

**Figure 5-28: PMPIDR0 bit assignments****RES0, [31:8]***res0*

Reserved.

**Part\_0, [7:0]****0x43**

Least significant byte of the performance monitor part number.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The PMPIDR0 can be accessed through the external debug interface, offset 0xFE0.

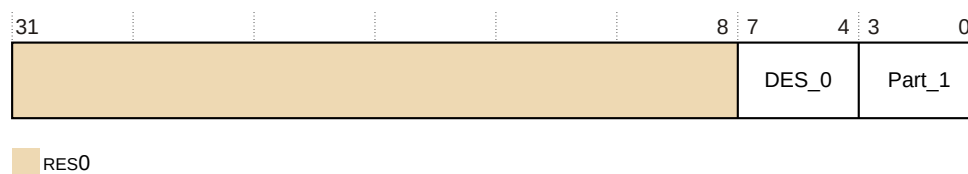
**5.4.8 PMPIDR1, Performance Monitors Peripheral Identification Register 1**

The PMPIDR1 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMPIDR1 is a 32-bit register.

**Figure 5-29: PMPIDR1 bit assignments**



RES0, [31:8]

*res0*

Reserved.

DES\_0, [7:4]

0xB

Arm Limited. This is the least significant nibble of JEP106 ID code.

Part\_1, [3:0]

0xD

Most significant nibble of the performance monitor part number.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The PMPIDR1 can be accessed through the external debug interface, offset 0xFE4.

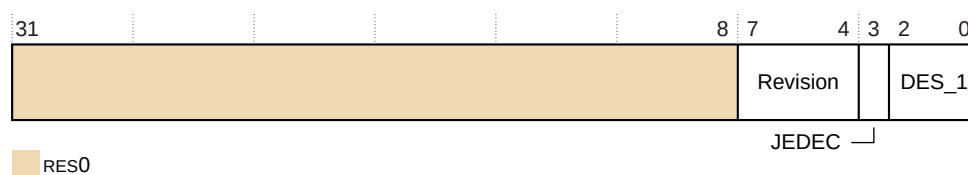
### 5.4.9 PMPIDR2, Performance Monitors Peripheral Identification Register 2

The PMPIDR2 provides information to identify a Performance Monitor component.

## Bit field descriptions

The PMPIDR2 is a 32-bit register.

**Figure 5-30: PMPIDR2 bit assignments**



**RES0, [31:8]**  
**res0**  
Reserved.

**Revision, [7:4]**  
**0x2**  
r1p1.

**JEDEC, [3]**  
**0b1**  
RAO. Indicates a JEP106 identity code is used.

**DES\_1, [2:0]**  
**0b011**  
Arm Limited. This is the most significant nibble of JEP106 ID code.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The PMPIDR2 can be accessed through the external debug interface, offset 0xFE8.

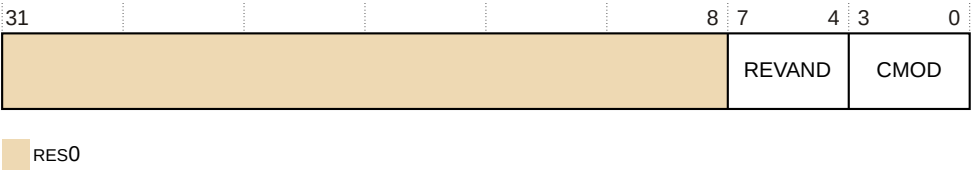
5.4.10 PMPIDR3, Performance Monitors Peripheral Identification Register 3

The PMPIDR3 provides information to identify a Performance Monitor component.

Bit field descriptions

The PMPIDR3 is a 32-bit register.

Figure 5-31: PMPIDR3 bit assignments



**RES0, [31:8]**  
**res0**  
Reserved.

REVAND, [7:4]

0x0  
Part minor revision.

CMOD, [3:0]

0x0  
Customer modified.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The PMPIDR3 can be accessed through the external debug interface, offset 0xFEC.

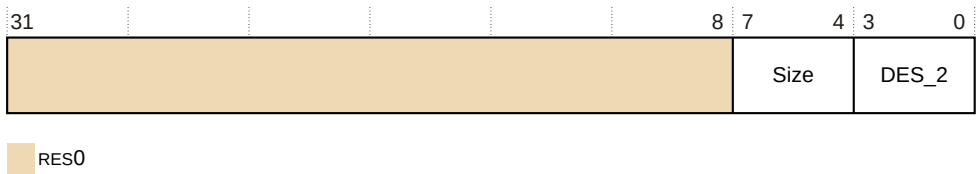
5.4.11 PMPIDR4, Performance Monitors Peripheral Identification Register 4

The PMPIDR4 provides information to identify a Performance Monitor component.

Bit field descriptions

The PMPIDR4 is a 32-bit register.

Figure 5-32: PMPIDR4 bit assignments



RES0, [31:8]

res0  
Reserved.

Size, [7:4]

0x0  
Size of the component. Log<sub>2</sub> the number of 4KB pages from the start of the component to the end of the component ID registers.

DES\_2, [3:0]

0x4  
Arm Limited. This is the least significant nibble JEP106 continuation code.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

The PMPIDR4 can be accessed through the external debug interface, offset 0xFD0.

## 5.4.12 PMPIDRn, Performance Monitors Peripheral Identification Register 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers.

They are reserved for future use and are **RES0**.

## 5.5 PMU snapshot registers

PMU snapshot registers are an **IMPLEMENTATION DEFINED** extension to an Arm®v8-A compliant PMU to support an external core monitor that connects to a system profiler.

### 5.5.1 PMU snapshot register summary

The snapshot registers are visible in an **IMPLEMENTATION DEFINED** region of the PMU external debug interface. Each time the debugger sends a snapshot request, information is collected to see how the code is executed in the different cores.

The following table describes the PMU snapshot registers implemented in the core.

**Table 5-7: PMU snapshot register summary**

Offset	Name	Type	Width	Description
0x600	PMPCSSR_LO	RO	32	5.5.2 PMPCSSR, PMU Snapshot Program Counter Sample Register on page 312
0x604	PMPCSSR_HI	RO	32	
0x608	PMCIDSSR	RO	32	5.5.3 PMCIDSSR, PMU Snapshot CONTEXTIDR_EL1 Sample Register on page 313
0x60C	PMCID2SSR	RO	32	5.5.4 PMCID2SSR, PMU Snapshot CONTEXTIDR_EL2 Sample Register on page 314
0x610	PMSSSR	RO	32	5.5.5 PMSSSR, PMU Snapshot Status Register on page 314
0x614	PMOVSSR	RO	32	5.5.6 PMOVSSR, PMU Snapshot Overflow Status Register on page 315
0x618	PMCCNTSR_LO	RO	32	5.5.7 PMCCNTSR, PMU Snapshot Cycle Counter Register on page 315
0x61C	PMCCNTSR_HI	RO	32	
0x620 + 4xn	PMEVCNTSRn	RO	32	5.5.8 PMEVCNTSRn, PMU Snapshot Cycle Counter Registers 0-5 on page 316
0x6F0	PMSSCR	WO	32	5.5.9 PMSSCR, PMU Snapshot Capture Register on page 316



## 5.5.2 PMPCSSR, PMU Snapshot Program Counter Sample Register

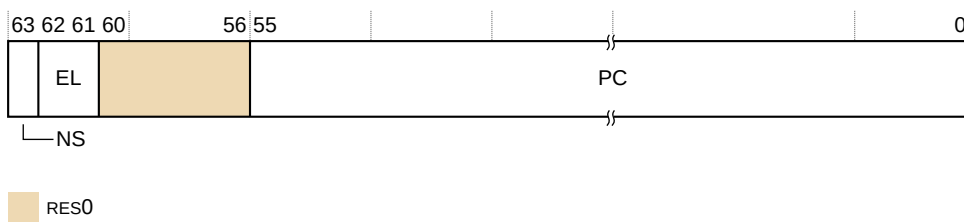
The PMPCSSR holds the same value as the PMPCSR register at the time of the snapshot.

However, unlike the other view of PMPCSR, it is not sensitive to reads. That is, reads of PMPCSSR through the PMU snapshot view do not cause a new sample capture and do not change PMCID1SR, PMCID2SR, or PMVIDSR.

### Bit field descriptions

The PMPCSSR is a 64-bit read-only register.

**Figure 5-33: PMPCSSR bit assignments**



#### NS, [63]

Non-secure sample.

#### EL, [62:61]

Exception level sample.

#### RES0, [60:56]

Reserved, **RES0**.

#### PC, [55:0]

Sampled PC.

### Configurations

There are no configuration notes.

### Usage constraints

Any access to PMPCSSR returns an error if any of the following occurs:

- The Core power domain is off.
- DoubleLockStatus() == TRUE.

### 5.5.3 PMCIDSSR, PMU Snapshot CONTEXTIDR\_EL1 Sample Register

The PMCIDSSR holds the same value as the PMCID1SR register at the time of the snapshot.

#### Configurations

There are no configuration notes.

#### Usage constraints

Any access to PMCIDSSR returns an error if any of the following occurs:

- The Core power domain is off.
- DoubleLockStatus() == TRUE.

### 5.5.4 PMCID2SSR, PMU Snapshot CONTEXTIDR\_EL2 Sample Register

The PMCID2SSR holds the same value as the PMCID2SR register at the time of the snapshot.

#### Configurations

There are no configuration notes.

#### Usage constraints

Any access to PMCID2SSR returns an error if any of the following occurs:

- The Core power domain is off.
- DoubleLockStatus() == TRUE.

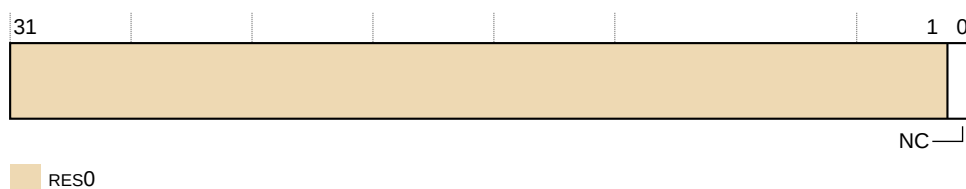
### 5.5.5 PMSSSR, PMU Snapshot Status Register

The PMSSSR holds status information about the captured counters.

#### Bit field descriptions

The PMSSSR is a 32-bit read-only register.

**Figure 5-34: PMSSSR bit assignments**



#### RES0, [31:1]

Reserved, RES0.

**NC, [0]**

No capture. This bit indicates whether the PMU counters have been captured. The possible values are:

0	PMU counters are captured.
1	PMU counters are not captured.

If there is a security violation, the core does not capture the event counters. The external monitor is responsible for keeping track of whether it managed to capture the snapshot registers from the core.

This bit does not reflect the status of the captured Program Counter Sample registers.

The core resets this bit to 1 by a Warm reset but MPSSSR.NC is overwritten at the first capture.

**Configurations**

There are no configuration notes.

**Usage constraints**

Any access to PMSSSR returns an error if any of the following occurs:

- The Core power domain is off.
- DoubleLockStatus() == TRUE.

## 5.5.6 PMOVSSR, PMU Snapshot Overflow Status Register

The PMOVSSR is a captured copy of PMOVSR.

Once it is captured, the value in PMOVSSR is unaffected by writes to PMOVSET\_ELO and PMOVCLR\_ELO.

**Configurations**

There are no configuration notes.

**Usage constraints**

Any access to PMOVSSR returns an error if any of the following occurs:

- The Core power domain is off.
- DoubleLockStatus() == TRUE.

### 5.5.7 PMCCNTSR, PMU Snapshot Cycle Counter Register

The PMCCNTSR is a captured copy of PMCCNTR\_ELO.

Once it is captured, the value in PMCCNTSR is unaffected by writes to PMCCNTR\_ELO and PMCR\_ELO.C.

#### Configurations

There are no configuration notes.

#### Usage constraints

Any access to PMCCNTSR returns an error if any of the following occurs:

- The Core power domain is off.
- DoubleLockStatus() == TRUE.

### 5.5.8 PMEVCNTRn, PMU Snapshot Cycle Counter Registers 0-5

The PMEVCNTRn, are captured copies of PMEVCNTRn\_ELO, n is 0-5.

When they are captured, the value in PMSSEVCNTRn is unaffected by writes to PMSSEVCNTRn\_ELO and PMCR\_ELO.P.

#### Configurations

There are no configuration notes.

#### Usage constraints

Any access to PMSSEVCNTRn returns an error if any of the following occurs:

- The Core power domain is off.
- DoubleLockStatus() == TRUE.

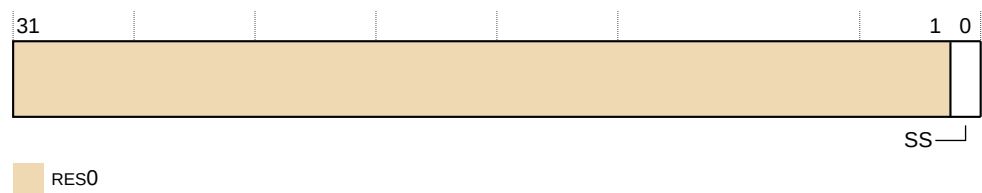
### 5.5.9 PMSSCR, PMU Snapshot Capture Register

The PMSSCR provides a mechanism for software to initiate a sample.

#### Bit field descriptions

The PMSSCR is a 32-bit write-only register.

Figure 5-35: PMSSCR bit assignments



**RES0, [31:1]**

Reserved, **RES0**.

**SS, [0]**

Capture now. The possible values are:

- |   |                                 |
|---|---------------------------------|
| 0 | <b>IGNORED.</b>                 |
| 1 | Initiate a capture immediately. |

**Configurations**

There are no configuration notes.

**Usage constraints**

Any access to PMSSCR returns an error if any of the following occurs:

- The Core power domain is off.
- DoubleLockStatus() == TRUE.

## 5.6 ETM registers

This chapter describes the ETM registers.

### 5.6.1 ETM register summary

This section summarizes the ETM trace unit registers.

All ETM trace unit registers are 32-bit wide. The description of each register includes its offset from a base address. The base address is defined by the system integrator when placing the ETM trace unit in the Debug-APB memory map.

The following table lists all of the ETM trace unit registers.

Table 5-8: ETM trace unit register summary

Offset	Name	Type	Reset	Description
0x004	TRCPRGCTLR	RW	0x00000000	<a href="#">5.6.59 TRCPRGCTLR, Programming Control Register</a> on page 372

Offset	Name	Type	Reset	Description
0x00C	TRCSTATR	RO	0x00000003	<a href="#">5.6.67 TRCSTATR, Status Register on page 381</a>
0x010	TRCCONFIGR	RW	UNK	<a href="#">5.6.20 TRCCONFIGR, Trace Configuration Register on page 337</a>
0x018	TRCAUXCTLR	RW	0x00000000	<a href="#">5.6.5 TRCAUXCTLR, Auxiliary Control Register on page 323</a>
0x020	TRCEVENTCTLOR	RW	UNK	<a href="#">5.6.26 TRCEVENTCTLOR, Event Control 0 Register on page 342</a>
0x024	TRCEVENTCTL1R	RW	UNK	<a href="#">5.6.27 TRCEVENTCTL1R, Event Control 1 Register on page 344</a>
0x02C	TRCSTALLCTLR	RW	UNK	<a href="#">5.6.66 TRCSTALLCTLR, Stall Control Register on page 380</a>
0x030	TRCTSCTLR	RW	UNK	<a href="#">5.6.70 TRCTSCTLR, Global Timestamp Control Register on page 383</a>
0x034	TRCSYNCPR	RW	UNK	<a href="#">5.6.68 TRCSYNCPR, Synchronization Period Register on page 381</a>
0x038	TRCCCCTLR	RW	UNK	<a href="#">5.6.7 TRCCCCTLR, Cycle Count Control Register on page 326</a>
0x03C	TRCBBCTLR	RW	UNK	<a href="#">5.6.6 TRCBBCTLR, Branch Broadcast Control Register on page 325</a>
0x040	TRCTRACEIDR	RW	UNK	<a href="#">5.6.69 TRCTRACEIDR, Trace ID Register on page 382</a>
0x080	TRCVICTLR	RW	UNK	<a href="#">5.6.71 TRCVICTLR, ViewInst Main Control Register on page 384</a>
0x084	TRCVIICTLR	RW	UNK	<a href="#">5.6.72 TRCVIICTLR, ViewInst Include-Exclude Control Register on page 386</a>
0x088	TRCVISSCTLR	RW	UNK	<a href="#">5.6.73 TRCVISSCTLR, ViewInst Start-Stop Control Register on page 387</a>
0x100	TRCSEQEVR0	RW	UNK	<a href="#">5.6.61 TRCSEQEVRn, Sequencer State Transition Control Registers 0-2 on page 374</a>
0x104	TRCSEQEVR1	RW	UNK	<a href="#">5.6.61 TRCSEQEVRn, Sequencer State Transition Control Registers 0-2 on page 374</a>
0x108	TRCSEQEVR2	RW	UNK	<a href="#">5.6.61 TRCSEQEVRn, Sequencer State Transition Control Registers 0-2 on page 374</a>
0x118	TRCSEQRSTEV	RW	UNK	<a href="#">5.6.62 TRCSEQRSTEV, Sequencer Reset Control Register on page 376</a>
0x11C	TRCSEQSTR	RW	UNK	<a href="#">5.6.63 TRCSEQSTR, Sequencer State Register on page 377</a>
0x120	TRCEXTINSEL	RW	UNK	<a href="#">5.6.28 TRCEXTINSEL, External Input Select Register on page 345</a>
0x140	TRCCNTRLDVRO	RW	UNK	<a href="#">5.6.18 TRCCNTRLDVRn, Counter Reload Value Registers 0-1 on page 335</a>
0x144	TRCCNTRLDVR1	RW	UNK	<a href="#">5.6.18 TRCCNTRLDVRn, Counter Reload Value Registers 0-1 on page 335</a>
0x150	TRCCNTCTLR0	RW	UNK	<a href="#">5.6.16 TRCCNTCTLR0, Counter Control Register 0 on page 332</a>
0x154	TRCCNTCTLR1	RW	UNK	<a href="#">5.6.17 TRCCNTCTLR1, Counter Control Register 1 on page 334</a>
0x160	TRCCNTVR0	RW	UNK	<a href="#">5.6.19 TRCCNTVRn, Counter Value Registers 0-1 on page 336</a>
0x164	TRCCNTVR1	RW	UNK	<a href="#">5.6.19 TRCCNTVRn, Counter Value Registers 0-1 on page 336</a>
0x180	TRCIDR8	RO	0x00000000	<a href="#">5.6.35 TRCIDR8, ID Register 8 on page 356</a>
0x184	TRCIDR9	RO	0x00000000	<a href="#">5.6.36 TRCIDR9, ID Register 9 on page 357</a>
0x188	TRCIDR10	RO	0x00000000	<a href="#">5.6.37 TRCIDR10, ID Register 10 on page 357</a>
0x18C	TRCIDR11	RO	0x00000000	<a href="#">5.6.38 TRCIDR11, ID Register 11 on page 358</a>
0x190	TRCIDR12	RO	0x00000000	<a href="#">5.6.39 TRCIDR12, ID Register 12 on page 358</a>
0x194	TRCIDR13	RO	0x00000000	<a href="#">5.6.40 TRCIDR13, ID Register 13 on page 359</a>
0x1C0	TRCIMSPEC0	RW	0x00000000	<a href="#">5.6.41 TRCIMSPEC0, Implementation Specific Register 0 on page 359</a>
0x1E0	TRCIDR0	RO	0x28000EA1	<a href="#">5.6.29 TRCIDR0, ID Register 0 on page 346</a>
0x1E4	TRCIDR1	RO	0x41001423	<a href="#">5.6.30 TRCIDR1, ID Register 1 on page 349</a>
0x1E8	TRCIDR2	RO	0x20001088	<a href="#">5.6.31 TRCIDR2, ID Register 2 on page 350</a>
0x1EC	TRCIDR3	RO	0x0D7B0004	<a href="#">5.6.32 TRCIDR3, ID Register 3 on page 351</a>
0x1F0	TRCIDR4	RO	0x11170004	<a href="#">5.6.33 TRCIDR4, ID Register 4 on page 353</a>

Offset	Name	Type	Reset	Description
0x1F4	TRCIDR5	RO	0x28C7082F	5.6.34 TRCIDR5, ID Register 5 on page 354
0x200	TRCRSCTLRn	RW	UNK	5.6.60 TRCRSCTLRn, Resource Selection Control Registers 2-15 on page 373, n is 2, 15
0x280	TRCSSCCR0	RW	UNK	5.6.64 TRCSSCCR0, Single-Shot Comparator Control Register 0 on page 378
0x2A0	TRCSSCSR0	RW	UNK	5.6.65 TRCSSCSR0, Single-Shot Comparator Status Register 0 on page 379
0x300	TRCOSLAR	WO	0x00000001	5.6.49 TRCOSLAR, OS Lock Access Register on page 365
0x304	TRCOSLSR	RO	0x0000000A	5.6.50 TRCOSLSR, OS Lock Status Register on page 366
0x310	TRCPDCR	RW	0x00000000	5.6.51 TRCPDCR, Power Down Control Register on page 367
0x314	TRCPDSR	RO	0x00000022	5.6.52 TRCPDSR, Power Down Status Register on page 368
0x400	TRCACVRn	RW	UNK	5.6.3 TRCACVRn, Address Comparator Value Registers 0-7 on page 321
0x480	TRCACATRn	RW	UNK	5.6.2 TRCACATRn, Address Comparator Access Type Registers 0-7 on page 320
0x600	TRCCIDCVR0	RW	UNK	5.6.9 TRCCIDCVR0, Context ID Comparator Value Register 0 on page 327
0x640	TRCVMIDCVR0	RW	UNK	5.6.75 TRCVMIDCVR0, VMID Comparator Value Register 0 on page 388
0x680	TRCCIDCCTLRO	RW	UNK	5.6.8 TRCCIDCCTLRO, Context ID Comparator Control Register 0 on page 327
0x688	TRCVMIDCCTLRO	RW	UNK	5.6.74 TRCVMIDCCTLRO, Virtual context identifier Comparator Control Register 0 on page 388
0xEE4	TRCITATBIDR	RW	UNK	5.6.42 TRCITATBIDR, Integration ATB Identification Register on page 360
0xEEC	TRCITIDATAR	WO	UNK	5.6.46 TRCITIDATAR, Integration Instruction ATB Data Register on page 363
0xEF4	TRCITIATBINR	RO	UNK	5.6.44 TRCITIATBINR, Integration Instruction ATB In Register on page 361
0xEFC	TRCITIATBOUTr	WO	UNK	5.6.45 TRCITIATBOUTr, Integration Instruction ATB Out Register on page 362
0xF00	TRCITCTRL	RW	0x00000000	5.6.43 TRCITCTRL, Integration Mode Control Register on page 360
0xFA0	TRCCLAIMSET	RW	UNK	5.6.15 TRCCLAIMSET, Claim Tag Set Register on page 331
0xFA4	TRCCLAIMCLR	RW	0x00000000	5.6.14 TRCCLAIMCLR, Claim Tag Clear Register on page 331
0xFA8	TRCDEVAFF0	RO	UNK	5.6.21 TRCDEVAFF0, Device Affinity Register 0 on page 340
0xFAC	TRCDEVAFF1	RO	UNK	5.6.22 TRCDEVAFF1, Device Affinity Register 1 on page 340
0xFB0	TRCLAR	WO	UNK	5.6.47 TRCLAR, Software Lock Access Register on page 364
0xFB4	TRCLSR	RO	0x00000000	5.6.48 TRCLSR, Software Lock Status Register on page 364
0xFB8	TRCAUTHSTATUS	RO	UNK	5.6.4 TRCAUTHSTATUS, Authentication Status Register on page 322
0xFBC	TRCDEVARCH	RO	0x47724A13	5.6.23 TRCDEVARCH, Device Architecture Register on page 340
0xFC8	TRCDEVID	RO	0x00000000	5.6.24 TRCDEVID, Device ID Register on page 341
0xFCC	TRCDEVTYPE	RO	0x00000013	5.6.25 TRCDEVTYPE, Device Type Register on page 341
0xFE0	TRCPIDR0	RO		5.6.53 TRCPIDR0, ETM Peripheral Identification Register 0 on page 369
0xFE4	TRCPIDR1	RO	0x000000BD	5.6.54 TRCPIDR1, ETM Peripheral Identification Register 1 on page 369
0xFE8	TRCPIDR2	RO	0x0200001B	5.6.55 TRCPIDR2, ETM Peripheral Identification Register 2 on page 370
0xFEC	TRCPIDR3	RO	0x00000000	5.6.56 TRCPIDR3, ETM Peripheral Identification Register 3 on page 371
0xFD0	TRCPIDR4	RO	0x00000004	5.6.57 TRCPIDR4, ETM Peripheral Identification Register 4 on page 372
0xFD4-0xFDF	TRCPIDRn	RO	0x00000000	5.6.58 TRCPIDRn, ETM Peripheral Identification Registers 5-7 on page 372
0xFF0	TRCCIDR0	RO	0x0000000D	5.6.10 TRCCIDR0, ETM Component Identification Register 0 on page 328
0xFF4	TRCCIDR1	RO	0x00000090	5.6.11 TRCCIDR1, ETM Component Identification Register 1 on page 329
0xFF8	TRCCIDR2	RO	0x00000005	5.6.12 TRCCIDR2, ETM Component Identification Register 2 on page 329

Offset	Name	Type	Reset	Description
0xFFC	TRCCIDR3	RO	0x000000B1	5.6.13 TRCCIDR3, ETM Component Identification Register 3 on page 330

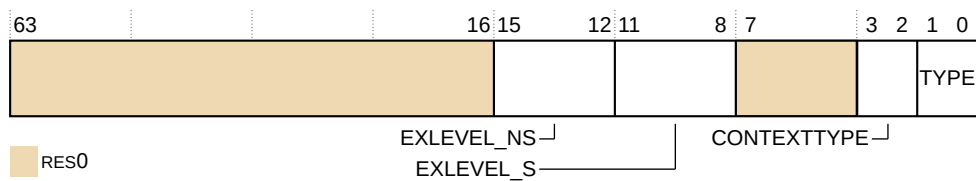
## 5.6.2 TRCACATR<sub>n</sub>, Address Comparator Access Type Registers 0-7

The TRCACATR<sub>n</sub> registers control the access for the corresponding address comparators.

### Bit field descriptions

The TRCACATR<sub>n</sub> registers are 64-bit registers.

**Figure 5-36: TRCACATR<sub>n</sub> bit assignments**



### RES0, [63:16]

RES0      Reserved

### EXLEVEL\_NS, [15:12]

Each bit controls whether a comparison can occur in Non-secure state for the corresponding Exception level. The possible values are:

- |   |   |
|---|---|
| 0 | The trace unit can perform a comparison, in Non-secure state, for Exception level <i>n</i> .      |
| 1 | The trace unit does not perform a comparison, in Non-secure state, for Exception level <i>n</i> . |

The Exception levels are:

- |         |                    |
|---------|--------------------|
| Bit[12] | Exception level 0  |
| Bit[13] | Exception level 1  |
| Bit[14] | Exception level 2  |
| Bit[15] | Always <b>RES0</b> |

### EXLEVEL\_S, [11:8]

Each bit controls whether a comparison can occur in Secure state for the corresponding Exception level. The possible values are:

- |   |   |
|---|---|
| 0 | The trace unit can perform a comparison, in Secure state, for Exception level <i>n</i> .      |
| 1 | The trace unit does not perform a comparison, in Secure state, for Exception level <i>n</i> . |



The Exception levels are:

Bit[8]	Exception level 0
Bit[9]	Exception level 1
Bit[10]	Always <b>RES0</b>
Bit[11]	Exception level 3

#### **RES0, [7:4]**

<b>RES0</b>	Reserved
-------------	----------

#### **CONTEXT TYPE, [3:2]**

Controls whether the trace unit performs a Context ID comparison, a VMID comparison, or both comparisons:

0b00	The trace unit does not perform a Context ID comparison.
0b01	The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.
0b10	The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.
0b11	The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

#### **TYPE, [1:0]**

Type of comparison:

0b00	Instruction address, <b>RES0</b>
------	----------------------------------

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCACATRn registers can be accessed through the external debug interface, offset 0x480–0x4B8.

### **5.6.3 TRCACVRn, Address Comparator Value Registers 0-7**

The TRCACVRn registers indicate the address for the address comparators.

#### **Bit field descriptions**

The TRCACVRn registers are 64-bit registers.

Figure 5-37: TRCACVRn bit assignments



ADDRESS, [63:0]

The address value to compare against

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCACVRn can be accessed through the external debug interface, offset 0x400-0x43C.

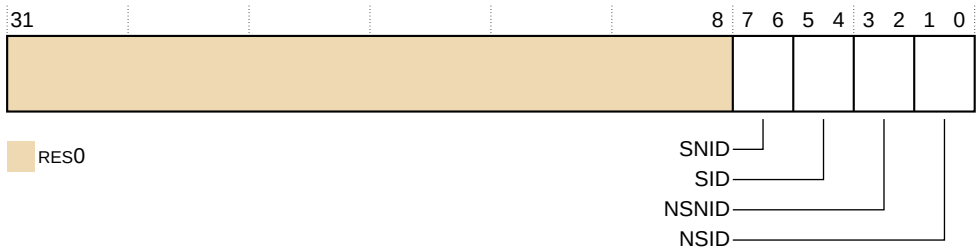
5.6.4 TRCAUTHSTATUS, Authentication Status Register

The TRCAUTHSTATUS indicates the current level of tracing permitted by the system.

Bit field descriptions

The TRCAUTHSTATUS is a 32-bit register.

Figure 5-38: TRCAUTHSTATUS bit assignments



RES0, [31:8]

RES0      Reserved.

SNID, [7:6]

Secure Non-invasive Debug:

0b10	Secure Non-invasive Debug implemented but disabled.
0b11	Secure Non-invasive Debug implemented and enabled.

SID, [5:4]

Secure Invasive Debug:

0b00	Secure Invasive Debug is not implemented.
------	---

**NSNID, [3:2]**

Non-secure Non-invasive Debug:

- |      |  |
|------|--|
| 0b10 | Non-secure Non-invasive Debug implemented but disabled, <b>NIDEN</b> =0. |
| 0b11 | Non-secure Non-invasive Debug implemented and enabled, <b>NIDEN</b> =1.  |

**NSID, [1:0]**

Non-secure Invasive Debug:

- |      |   |
|------|---|
| 0b00 | Non-secure Invasive Debug is not implemented. |
|------|---|

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCAUTHSTATUS can be accessed through the external debug interface, offset 0xFB8.

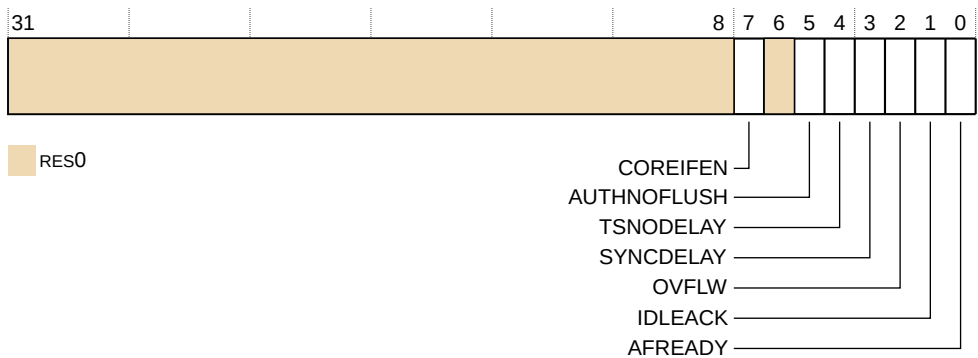
5.6.5 TRCAUXCTLR, Auxiliary Control Register

The TRCAUXCTLR provides **IMPLEMENTATION DEFINED** configuration and control options.

**Bit field descriptions**

The TRCAUXCTLR is a 32-bit register.

**Figure 5-39: TRCAUXCTLR bit assignments**



**RES0, [31:8]**

- |      |           |
|------|-----------|
| RES0 | Reserved. |
|------|-----------|

**COREIFEN, [7]**

Keep core interface enabled regardless of trace enable register state. The possible values are:

- |   |   |
|---|---|
| 0 | Core interface enabled is set by trace enable register state.     |
| 1 | Enable core interface, regardless of trace enable register state. |

**RES0, [6]**

**RES0** Reserved.

**AUTHNOFLUSH, [5]**

Do not flush trace on de-assertion of authentication inputs. The possible values are:

- |   |   |
|---|---|
| 0 | ETM trace unit FIFO is flushed and ETM trace unit enters idle state when <b>DBGEN</b> or <b>NIDEN</b> is LOW.             |
| 1 | ETM trace unit FIFO is not flushed and ETM trace unit does not enter idle state when <b>DBGEN</b> or <b>NIDEN</b> is LOW. |

When this bit is set to 1, the trace unit behavior deviates from architecturally-specified behavior.

**TSNODELAY, [4]**

Do not delay timestamp insertion based on FIFO depth. The possible values are:

- |   |   |
|---|---|
| 0 | Timestamp packets are inserted into FIFO only when trace activity is LOW. |
| 1 | Timestamp packets are inserted into FIFO irrespective of trace activity.  |

**SYNCDELAY, [3]**

Delay periodic synchronization if FIFO is more than half-full. The possible values are:

- |   |  |
|---|--|
| 0 | SYNC packets are inserted into FIFO only when trace activity is low. |
| 1 | SYNC packets are inserted into FIFO irrespective of trace activity.  |

**OVFLW, [2]**

Force overflow if synchronization is not completed when second synchronization becomes due. The possible values are:

- |   |   |
|---|---|
| 0 | No FIFO overflow when SYNC packets are delayed.     |
| 1 | Forces FIFO overflow when SYNC packets are delayed. |

When this bit is set to 1, the trace unit behavior deviates from architecturally-specified behavior.

**IDLEACK, [1]**

Force idle-drain acknowledge high, CPU does not wait for trace to drain before entering WFX state. The possible values are:

- |   |  |
|---|--|
| 0 | ETM trace unit idle acknowledge is asserted only when the ETM trace unit is in idle state. |
| 1 | ETM trace unit idle acknowledge is asserted irrespective of the ETM trace unit idle state. |

When this bit is set to 1, trace unit behavior deviates from architecturally-specified behavior.

**AFREADY, [0]**

Always respond to AFREADY immediately. Does not have any interaction with FIFO draining, even in WFI state. The possible values are:

- |   |  |
|---|--|
| 0 | ETM trace unit <b>AFREADYM</b> output is asserted only when the ETM trace unit is in idle state or when all the trace bytes in FIFO before a flush request are output. |
| 1 | ETM trace unit <b>AFREADYM</b> output is always asserted HIGH. When this bit is set to 1, trace unit behavior deviates from architecturally-specified behavior.        |

The TRCAUXCTLR can be accessed through the external debug interface, offset 0x018.

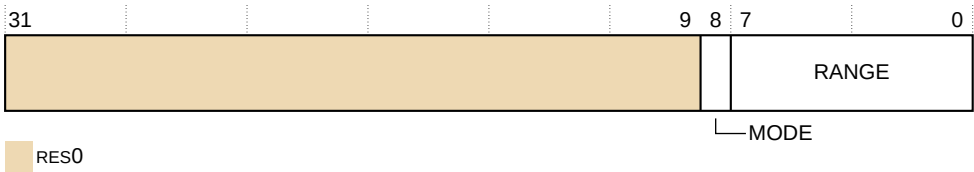
**5.6.6 TRCBBCTLR, Branch Broadcast Control Register**

The TRCBBCTLR controls how branch broadcasting behaves, and allows branch broadcasting to be enabled for certain memory regions.

**Bit field descriptions**

The TRCBBCTLR is a 32-bit register.

**Figure 5-40: TRCBBCTLR bit assignments**



**RES0, [31:9]**

- |      |          |
|------|----------|
| RES0 | Reserved |
|------|----------|

**MODE, [8]**

Mode bit:

- |   |   |
|---|---|
| 0 | Exclude mode. Branch broadcasting is not enabled in the address range that RANGE defines. |
| 1 | Include mode. Branch broadcasting is enabled in the address range that RANGE defines.     |

If RANGE==0 then the behavior of the trace unit is **CONSTRAINED UNPREDICTABLE**. That is, the trace unit might or might not consider any instructions to be in a branch broadcast region.

### RANGE, [7:0]

Address range field.

Selects which address range comparator pairs are in use with branch broadcasting. Each bit represents an address range comparator pair, so bit[*n*] controls the selection of address range comparator pair *n*. If bit[*n*] is:

- |   |   |
|---|---|
| 0 | The address range that address range comparator pair <i>n</i> defines, is not selected. |
| 1 | The address range that address range comparator pair <i>n</i> defines, is selected.     |

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCBBCTLR can be accessed through the external debug interface, offset 0x03C.

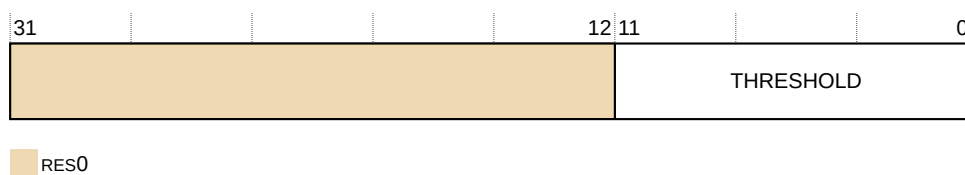
## 5.6.7 TRCCCCTLR, Cycle Count Control Register

The TRCCCCTLR sets the threshold value for cycle counting.

### Bit field descriptions

The TRCCCCTLR is a 32-bit register.

**Figure 5-41: TRCCCCTLR bit assignments**



### RES0, [31:12]

**RES0** Reserved.

### THRESHOLD, [11:0]

Instruction trace cycle count threshold.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCCTLR can be accessed through the external debug interface, offset 0x038.

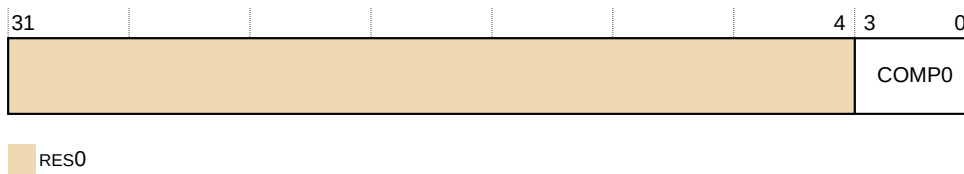
### 5.6.8 TRCCIDCTLR0, Context ID Comparator Control Register 0

The TRCCIDCTLR0 controls the mask value for the context ID comparators.

#### Bit field descriptions

The TRCCIDCTLR0 is a 32-bit register.

**Figure 5-42: TRCCIDCTLR0 bit assignments**



#### RES0, [31:4]

**RES0** Reserved.

#### COMPO, [3:0]

Controls the mask value that the trace unit applies to TRCCIDCVR0. Each bit in this field corresponds to a byte in TRCCIDCVR0. When a bit is:

- |   |   |
|---|---|
| 0 | The trace unit includes the relevant byte in TRCCIDCVR0 when it performs the Context ID comparison. |
| 1 | The trace unit ignores the relevant byte in TRCCIDCVR0 when it performs the Context ID comparison.  |

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

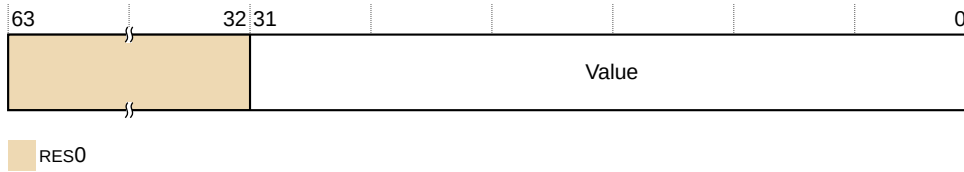
The TRCCIDCTLR0 can be accessed through the external debug interface, offset 0x680.

### 5.6.9 TRCCIDCVR0, Context ID Comparator Value Register 0

The TRCCIDCVR0 contains a Context ID value.

#### Bit field descriptions

The TRCCIDCVR0 is a 64-bit register.

**Figure 5-43: TRCCIDCVR0 bit assignments****RES0, [63:32]**

**RES0** Reserved.

**VALUE, [31:0]**

The data value to compare against.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

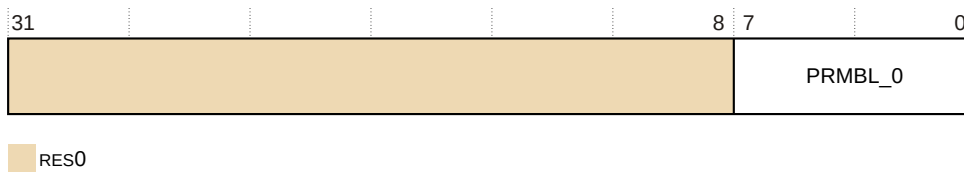
The TRCCIDCVR0 can be accessed through the external debug interface, offset 0x600.

**5.6.10 TRCCIDR0, ETM Component Identification Register 0**

The TRCCIDR0 provides information to identify a trace component.

**Bit field descriptions**

The TRCCIDR0 is a 32-bit register.

**Figure 5-44: TRCCIDR0 bit assignments****RES0, [31:8]**

**RES0** Reserved.

**PRMBL\_0, [7:0]**

0x0D Preamble byte 0.



Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCIDR0 can be accessed through the external debug interface, offset 0xFF0.

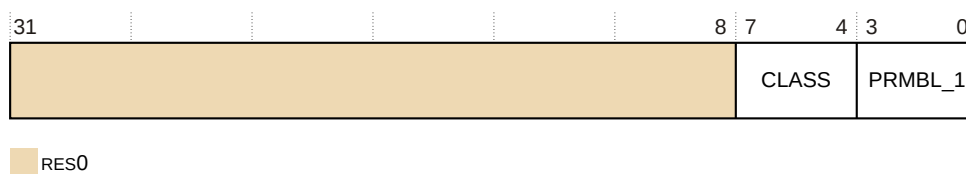
### 5.6.11 TRCCIDR1, ETM Component Identification Register 1

The TRCCIDR1 provides information to identify a trace component.

#### Bit field descriptions

The TRCCIDR1 is a 32-bit register.

**Figure 5-45: TRCCIDR1 bit assignments**



#### RES0, [31:8]

RES0      Reserved.

#### CLASS, [7:4]

0x9      Debug component.

#### PRMBL\_1, [3:0]

0x0      Preamble byte 1.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

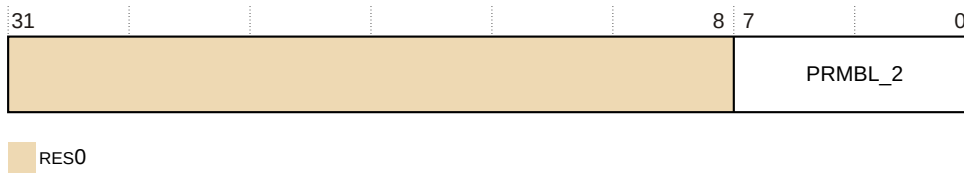
The TRCCIDR1 can be accessed through the external debug interface, offset 0xFF4.

### 5.6.12 TRCCIDR2, ETM Component Identification Register 2

The TRCCIDR2 provides information to identify a CTI component.

#### Bit field descriptions

The TRCCIDR2 is a 32-bit register.

**Figure 5-46: TRCCIDR2 bit assignments****RES0, [31:8]**

<b>RES0</b>	Reserved.
-------------	-----------

**PRMBL\_2, [7:0]**

0x05	Preamble byte 2.
------	------------------

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

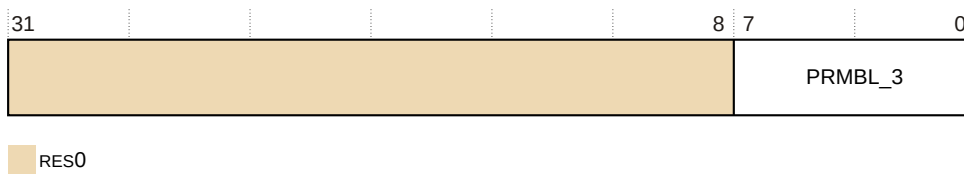
The TRCCIDR2 can be accessed through the external debug interface, offset 0xFF8.

### 5.6.13 TRCCIDR3, ETM Component Identification Register 3

The TRCCIDR3 provides information to identify a trace component.

**Bit field descriptions**

The TRCCIDR3 is a 32-bit register.

**Figure 5-47: TRCCIDR3 bit assignments****RES0, [31:8]**

<b>RES0</b>	Reserved.
-------------	-----------

**PRMBL\_3, [7:0]**

0xB1	Preamble byte 3.
------	------------------

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCIDR3 can be accessed through the external debug interface, offset 0xFFC.

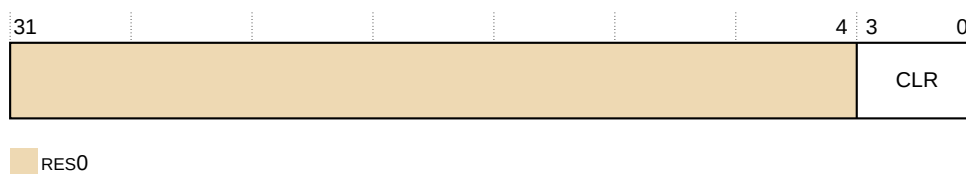
### 5.6.14 TRCCLAIMCLR, Claim Tag Clear Register

The TRCCLAIMCLR clears bits in the claim tag and determines the current value of the claim tag.

#### Bit field descriptions

The TRCCLAIMCLR is a 32-bit register.

**Figure 5-48: TRCCLAIMCLR bit assignments**



#### RES0, [31:4]

**RES0**      Reserved.

#### CLR, [3:0]

On reads, for each bit:

0	Claim tag bit is not set.
1	Claim tag bit is set.

On writes, for each bit:

0	Has no effect.
1	Clears the relevant bit of the claim tag.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCLAIMCLR can be accessed through the external debug interface, offset 0xFA4.

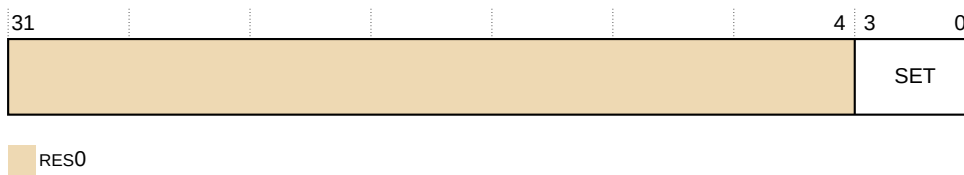
### 5.6.15 TRCCLAIMSET, Claim Tag Set Register

The TRCCLAIMSET sets bits in the claim tag and determines the number of claim tag bits implemented.

#### Bit field descriptions

The TRCCLAIMSET is a 32-bit register.

**Figure 5-49: TRCCLAIMSET bit assignments**



#### RES0, [31:4]

**RES0** Reserved.

#### SET, [3:0]

On reads, for each bit:

0	Claim tag bit is not implemented.
1	Claim tag bit is implemented.

On writes, for each bit:

0	Has no effect.
1	Sets the relevant bit of the claim tag.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

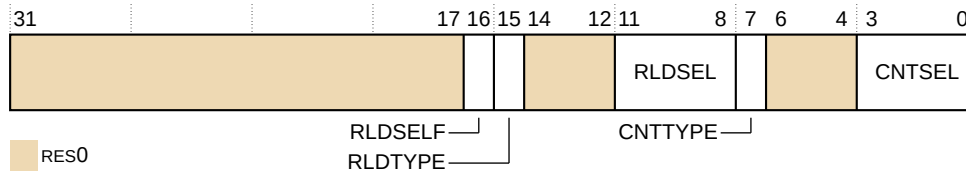
The TRCCLAIMSET can be accessed through the external debug interface, offset 0xFA0.

### 5.6.16 TRCCNTCTLR0, Counter Control Register 0

The TRCCNTCTLR0 controls the counter.

#### Bit field descriptions

The TRCCNTCTLR0 is a 32-bit register.

**Figure 5-50: TRCCNTCTLR0 bit assignments****RES0, [31:17]**

**RES0** Reserved.

**RLDSELF, [16]**

Defines whether the counter reloads when it reaches zero:

- |   |  |
|---|--|
| 0 | The counter does not reload when it reaches zero. The counter only reloads based on RLDTYPE and RLDSEL.  |
| 1 | The counter reloads when it reaches zero and the resource selected by CNTTYPE and CNTSEL is also active. The counter also reloads based on RLDTYPE and RLDSEL. |

**RLDTYPE, [15]**

Selects the resource type for the reload:

- |   |                                 |
|---|---------------------------------|
| 0 | Single selected resource.       |
| 1 | Boolean combined resource pair. |

**RES0, [14:12]**

**RES0** Reserved.

**RLDSEL, [11:8]**

Selects the resource number, based on the value of RLDTYPE:

When RLDTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When RLDTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

**CNTTYPE, [7]**

Selects the resource type for the counter:

- |   |                                 |
|---|---------------------------------|
| 0 | Single selected resource.       |
| 1 | Boolean combined resource pair. |

**RES0, [6:4]**

**RES0** Reserved.

**CNTSEL, [3:0]**

Selects the resource number, based on the value of CNTTYPE:

When CNTTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When CNTTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCNTCTLR0 can be accessed through the external debug interface, offset 0x150.

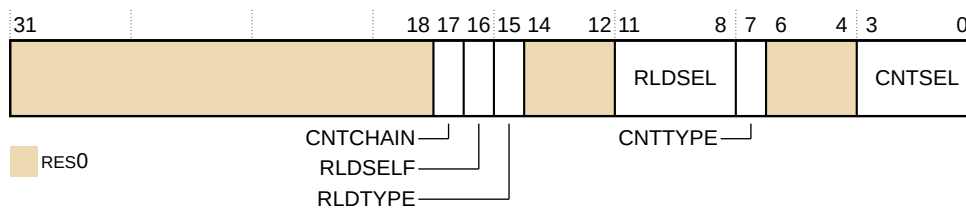
**5.6.17 TRCCNTCTLR1, Counter Control Register 1**

The TRCCNTCTLR1 controls the counter.

**Bit field descriptions**

The TRCCNTCTLR1 is a 32-bit register.

**Figure 5-51: TRCCNTCTLR1 bit assignments**

**RES0, [31:18]**

**RES0** Reserved.

**CNTCHAIN, [17]**

Defines whether the counter decrements when the counter reloads. This enables two counters to be used in combination to provide a larger counter:

- |   |  |
|---|--|
| 0 | The counter operates independently from the counter. The counter only decrements based on CNTTYPE and CNTSEL.                            |
| 1 | The counter decrements when the counter reloads. The counter also decrements when the resource selected by CNTTYPE and CNTSEL is active. |

**RLDSELF, [16]**

Defines whether the counter reloads when it reaches zero:

- |   |   |
|---|---|
| 0 | The counter does not reload when it reaches zero. The counter only reloads based on RLDTYPE and RLDSEL.   |
| 1 | The counter reloads when it is zero and the resource selected by CNTTYPE and CNTSEL is also active. The counter also reloads based on RLDTYPE and RLDSEL. |

**RLDTYPE, [15]**

Selects the resource type for the reload:

- |   |                                 |
|---|---------------------------------|
| 0 | Single selected resource.       |
| 1 | Boolean combined resource pair. |

**RES0, [14:12]**

**RES0** Reserved.

**RLDSEL, [11:8]**

Selects the resource number, based on the value of RLDTYPE:

When RLDTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When RLDTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

**CNTTYPE, [7]**

Selects the resource type for the counter:

- |   |                                 |
|---|---------------------------------|
| 0 | Single selected resource.       |
| 1 | Boolean combined resource pair. |

**RES0, [6:4]**

**RES0** Reserved.

**CNTSEL, [3:0]**

Selects the resource number, based on the value of CNTTYPE:

When CNTTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When CNTTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCNTCTLR1 can be accessed through the external debug interface, offset 0x154.

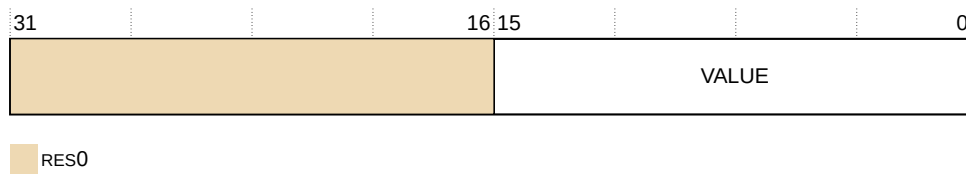
### 5.6.18 TRCCNTRLDVRn, Counter Reload Value Registers 0-1

The TRCCNTRLDVRn registers define the reload value for the counter.

#### Bit field descriptions

The TRCCNTRLDVRn registers are 32-bit registers.

**Figure 5-52: TRCCNTRLDVRn bit assignments**



#### RES0, [31:16]

RES0      Reserved

#### VALUE, [15:0]

Defines the reload value for the counter. This value is loaded into the counter each time the reload event occurs.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCNTRLDVRn registers can be accessed through the external debug interface, offsets:

#### TRCCNTRLDVR0

0x140

#### TRCCNTRLDVR1

0x144

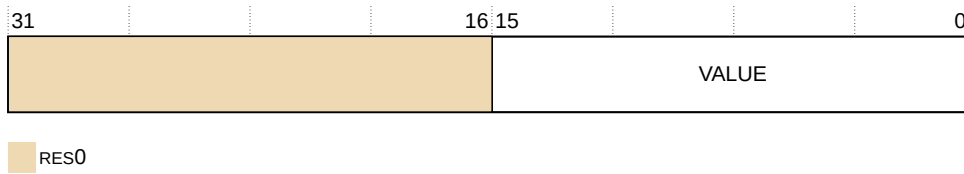
### 5.6.19 TRCCNTVRn, Counter Value Registers 0-1

The TRCCNTVRn registers contain the current counter value.

#### Bit field descriptions

The TRCCNTVRn registers are 32-bit registers.



**Figure 5-53: TRCCNTVRn bit assignments****RES0, [31:16]**

RES0      Reserved

**VALUE, [15:0]**

Contains the current counter value.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCNTVRn registers can be accessed through the external debug interface, offsets:

**TRCCNTVR0**

0x160

**TRCCNTVR1**

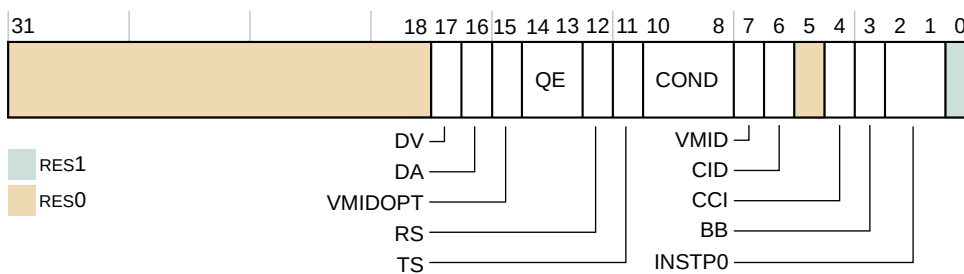
0x164

**5.6.20 TRCCONFIGR, Trace Configuration Register**

The TRCCONFIGR controls the tracing options.

**Bit field descriptions**

The TRCCONFIGR is a 32-bit register.

**Figure 5-54: TRCCONFIGR bit assignments**

**RES0, [31:18]**

<b>RES0</b>	Reserved.
-------------	-----------

**DV, [17]**

Enables data value tracing. The possible values are:

0	Disables data value tracing.
1	Enables data value tracing.

**DA, [16]**

Enables data address tracing. The possible values are:

0	Disables data address tracing.
1	Enables data address tracing.

**VMIDOPT, [15]**

Configures the Virtual context identifier value that is used by the trace unit, both for trace generation and in the Virtual context identifier comparators. The possible values are:

0b0	VTTBR_EL2.VMID is used. If the trace unit supports a Virtual context identifier larger than the VTTBR_EL2.VMID, the upper unused bits are always zero. If the trace unit supports a Virtual context identifier larger than 8 bits and if the VTCCR_EL2.VS bit forces use of an 8-bit Virtual context identifier, bits [15:8] of the trace unit Virtual context identifier are always zero.
0b1	CONTEXTIDR_EL2 is used. TRCIDR2.VMIDOPT indicates whether this field is implemented.

**QE, [14:13]**

Enables Q element. The possible values are:

0b00	Q elements are disabled.
0b01	Q elements with instruction counts are disabled. Q elements without instruction counts are disabled.
0b10	Reserved.
0b11	Q elements with and without instruction counts are enabled.

**RS, [12]**

Enables the return stack. The possible values are:

0	Disables the return stack.
1	Enables the return stack.

**TS, [11]**

Enables global timestamp tracing. The possible values are:

0	Disables global timestamp tracing.
1	Enables global timestamp tracing.

**COND, [10:8]**

Enables conditional instruction tracing. The possible values are:

0b000	Conditional instruction tracing is disabled.
0b001	Conditional load instructions are traced.
0b010	Conditional store instructions are traced.
0b011	Conditional load and store instructions are traced.
0b111	All conditional instructions are traced.

**VMID, [7]**

Enables VMID tracing. The possible values are:

0	Disables VMID tracing.
1	Enables VMID tracing.

**CID, [6]**

Enables context ID tracing. The possible values are:

0	Disables context ID tracing.
1	Enables context ID tracing.

**RES0, [5]**

<b>RES0</b>	Reserved.
-------------	-----------

**CCI, [4]**

Enables cycle counting instruction trace. The possible values are:

0	Disables cycle counting instruction trace.
1	Enables cycle counting instruction trace.

**BB, [3]**

Enables branch broadcast mode. The possible values are:

0	Disables branch broadcast mode.
1	Enables branch broadcast mode.

**INSTP0, [2:1]**

Controls whether load and store instructions are traced as P0 instructions. The possible values are:

0b00	Load and store instructions are not traced as P0 instructions.
0b01	Load instructions are traced as P0 instructions.
0b10	Store instructions are traced as P0 instructions.
0b11	Load and store instructions are traced as P0 instructions.

**RES1, [0]**

<b>RES1</b>	Reserved.
-------------	-----------

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCONFIGR can be accessed through the external debug interface, offset 0x010.

### 5.6.21 TRCDEVAFF0, Device Affinity Register 0

The TRCDEVAFF0 provides an additional core identification mechanism for scheduling purposes in a cluster. TRCDEVAFF0 is a read-only copy of MPIDR\_EL1[31:0] accessible from the external debug interface.

#### Bit field descriptions

The TRCDEVAFF0 is a 32-bit register and is a copy of MPIDR\_EL1[31:0]. See [3.1.65 MPIDR\\_EL1, Multiprocessor Affinity Register, EL1](#) on page 169 for full bit field descriptions.

### 5.6.22 TRCDEVAFF1, Device Affinity Register 1

The TRCDEVAFF1 provides an additional core identification mechanism for scheduling purposes in a cluster. TRCDEVAFF1 is a read-only copy of MPIDR\_EL1[63:32] accessible from the external debug interface.

#### Bit field descriptions

The TRCDEVAFF1 is a 32-bit register and is a copy of MPIDR\_EL1[63:32]. See [3.1.65 MPIDR\\_EL1, Multiprocessor Affinity Register, EL1](#) on page 169 for full bit field descriptions.

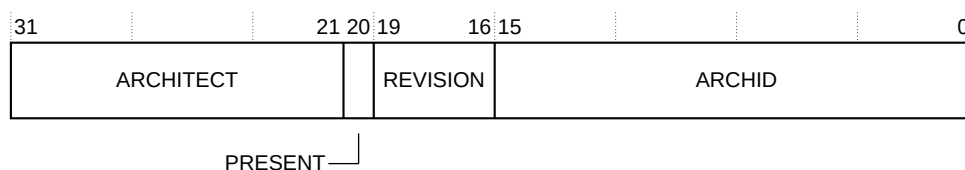
### 5.6.23 TRCDEVARCH, Device Architecture Register

The TRCDEVARCH identifies the ETM trace unit as an ETMv4 component.

#### Bit field descriptions

The TRCDEVARCH is a 32-bit register.

**Figure 5-55: TRCDEVARCH bit assignments**



#### ARCHITECT, [31:21]

Defines the architect of the component:

- 0x4      Arm JEP continuation.
- 0x3B    Arm JEP 106 code.

**PRESENT, [20]**

Indicates the presence of this register:

0b1          Register is present.

**REVISION, [19:16]**

Architecture revision:

0x02          Architecture revision 2.

**ARCHID, [15:0]**

Architecture ID:

0x4A13      ETMv4 component.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCDEVARCH can be accessed through the external debug interface, offset 0xFBC.

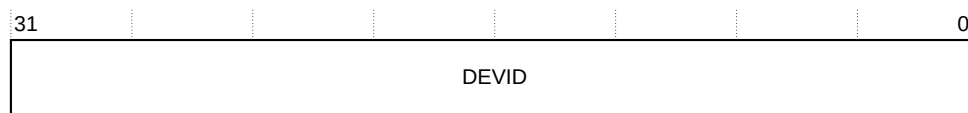
## 5.6.24 TRCDEVID, Device ID Register

The TRCDEVID indicates the capabilities of the ETM trace unit.

**Bit field descriptions**

The TRCDEVID is a 32-bit register.

**Figure 5-56: TRCDEVID bit assignments**

**DEVID, [31:0]**

RAZ. There are no component-defined capabilities.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCDEVID can be accessed through the external debug interface, offset 0xFC8.

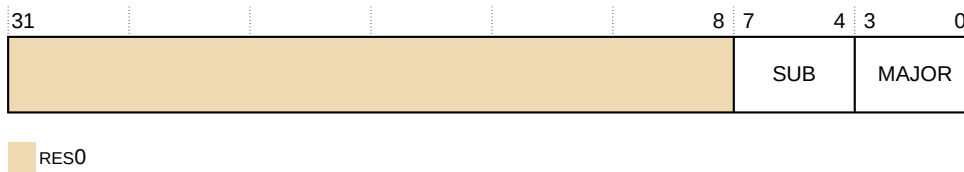
### 5.6.25 TRCDEVTYPE, Device Type Register

The TRCDEVTYPE indicates the type of the component.

#### Bit field descriptions

The TRCDEVTYPE is a 32-bit register.

**Figure 5-57: TRCDEVTYPE bit assignments**



#### RES0, [31:8]

**RES0**      Reserved.

#### SUB, [7:4]

The sub-type of the component:

0b0001      Core trace.

#### MAJOR, [3:0]

The main type of the component:

0b0011      Trace source.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

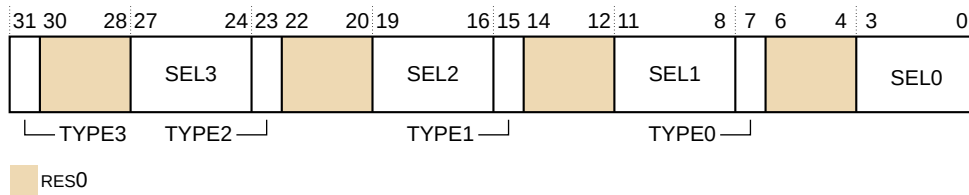
The TRCDEVTYPE can be accessed through the external debug interface, offset 0xFCC.

### 5.6.26 TRCEVENTCTL0R, Event Control 0 Register

The TRCEVENTCTL0R controls the tracing of events in the trace stream. The events also drive the external outputs from the ETM trace unit. The events are selected from the Resource Selectors.

#### Bit field descriptions

The TRCEVENTCTL0R is a 32-bit register.

**Figure 5-58: TRCEVENTCTL0R bit assignments****TYPE3, [31]**

Selects the resource type for trace event 3:

- 0 Single selected resource.
- 1 Boolean combined resource pair.

**RES0, [30:28]**

**RES0** Reserved.

**SEL3, [27:24]**

Selects the resource number, based on the value of TYPE3:

When TYPE3 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE3 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

**TYPE2, [23]**

Selects the resource type for trace event 2:

- 0 Single selected resource.
- 1 Boolean combined resource pair.

**RES0, [22:20]**

**RES0** Reserved.

**SEL2, [19:16]**

Selects the resource number, based on the value of TYPE2:

When TYPE2 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE2 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

**TYPE1, [15]**

Selects the resource type for trace event 1:

- 0 Single selected resource.
- 1 Boolean combined resource pair.

**RES0, [14:12]**

<b>RES0</b>	Reserved.
-------------	-----------

**SEL1, [11:8]**

Selects the resource number, based on the value of TYPE1:

When TYPE1 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE1 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

**TYPE0, [7]**

Selects the resource type for trace event 0:

0	Single selected resource.
1	Boolean combined resource pair.

**RES0, [6:4]**

<b>RES0</b>	Reserved.
-------------	-----------

**SEL0, [3:0]**

Selects the resource number, based on the value of TYPE0:

When TYPE0 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE0 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCEVENTCTL0R can be accessed through the external debug interface, offset 0x020.

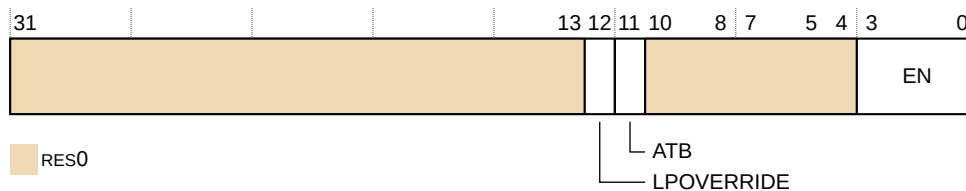
## 5.6.27 TRCEVENTCTL1R, Event Control 1 Register

The TRCEVENTCTL1R controls the behavior of the events that TRCEVENTCTL0R selects.

### Bit field descriptions

The TRCEVENTCTL1R is a 32-bit register.



**Figure 5-59: TRCEVENTCTL1R bit assignments****RES0, [31:13]**

**RES0**      Reserved.

**LPOVERRIDE, [12]**

Low-power state behavior override:

0	Low-power state behavior unaffected.
1	Low-power state behavior overridden. The resources and Event trace generation are unaffected by entry to a low-power state.

**ATB, [11]**

ATB trigger enable:

0	ATB trigger disabled.
1	ATB trigger enabled.

**RES0, [10:4]**

**RES0**      Reserved.

**EN, [3:0]**

One bit per event, to enable generation of an event element in the instruction trace stream when the selected event occurs:

0	Event does not cause an event element.
1	Event causes an event element.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCEVENTCTL1R can be accessed through the external debug interface, offset 0x024.

### 5.6.28 TRCEXTINSEL, External Input Select Register

The TRCEXTINSEL controls the selectors that choose an external input as a resource in the ETM trace unit. You can use the Resource Selectors to access these external input resources.

#### Bit field descriptions

The TRCEXTINSEL is a 32-bit register.

#### TRCEXTINSEL bit assignments

31	30	29				24	23	22	21				16	15	14	13				8	7	6	5				0								
			SEL3									SEL2									SEL1									SEL0					

RES0

#### RES0, [31:30]

RES0 Reserved.

#### SEL3, [29:24]

Selects an event from the external input bus for External Input Resource 3.

#### RES0, [23:22]

RES0 Reserved.

#### SEL2, [21:16]

Selects an event from the external input bus for External Input Resource 2.

#### RES0, [15:14]

RES0 Reserved.

#### SEL1, [13:8]

Selects an event from the external input bus for External Input Resource 1.

#### RES0, [7:6]

RES0 Reserved.

#### SEL0, [5:0]

Selects an event from the external input bus for External Input Resource 0.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCEXTINSEL can be accessed through the external debug interface, offset 0x120.

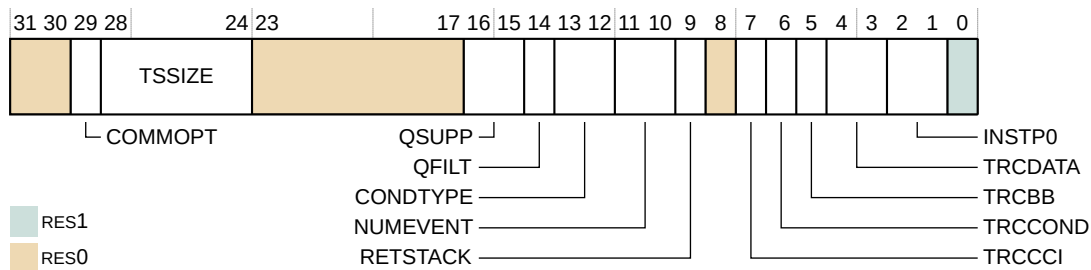
## 5.6.29 TRCIDR0, ID Register 0

The TRCIDR0 returns the tracing capabilities of the ETM trace unit.

### Bit field descriptions

The TRCIDR0 is a 32-bit register.

**Figure 5-60: TRCIDR0 bit assignments**



### RES0, [31:30]

RES0 Reserved.

### COMMOPT, [29]

Indicates the meaning of the commit field in some packets:

1 Commit mode 1.

### TSSIZE, [28:24]

Global timestamp size field:

0b01000 Implementation supports a maximum global timestamp of 64 bits.

### RES0, [23:17]

RES0 Reserved.

### QSUPP, [16:15]

Indicates Q element support:

0b00 Q elements not supported.

### QFILT, [14]

Indicates Q element filtering support:

0b0 Q element filtering not supported.

### CONDTYPE, [13:12]

Indicates how conditional results are traced:

0b00 Conditional trace not supported.

#### NUMEVENT, [11:10]

Number of events supported in the trace, minus 1:

0b11 Four events supported.

#### RETSTACK, [9]

Return stack support:

1 Return stack implemented.

#### RES0, [8]

RES0 Reserved.

#### TRCCCI, [7]

Support for cycle counting in the instruction trace:

1 Cycle counting in the instruction trace is implemented.

#### TRCCOND, [6]

Support for conditional instruction tracing:

0 Conditional instruction tracing is not supported.

#### TRCBB, [5]

Support for branch broadcast tracing:

1 Branch broadcast tracing is implemented.

#### TRCDATA, [4:3]

Conditional tracing field:

0b00 Tracing of data addresses and data values is not implemented.

#### INSTP0, [2:1]

P0 tracing support field:

0b00 Tracing of load and store instructions as P0 elements is not supported.

#### RES1, [0]

RES1 Reserved.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR0 can be accessed through the external debug interface, offset 0x1E0.

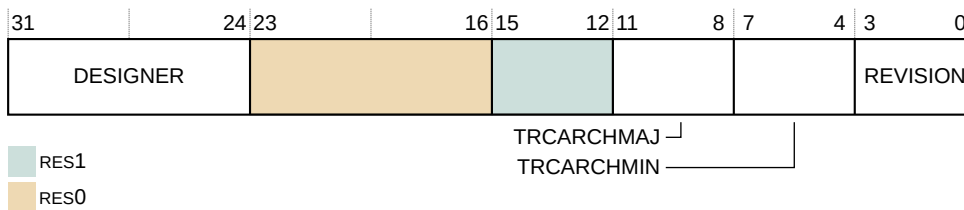
### 5.6.30 TRCIDR1, ID Register 1

The TRCIDR1 returns the base architecture of the trace unit.

#### Bit field descriptions

The TRCIDR1 is a 32-bit register.

**Figure 5-61: TRCIDR1 bit assignments**



#### DESIGNER, [31:24]

Indicates which company designed the trace unit:

0x41 Arm.

#### RES0, [23:16]

RES0 Reserved.

#### RES1, [15:12]

RES1 Reserved.

#### TRCARCHMAJ, [11:8]

Major trace unit architecture version number:

0x4 ETMv4.

#### TRCARCHMIN, [7:4]

Minor trace unit architecture version number:

0x2 ETMv4.2

#### REVISION, [3:0]

Trace unit implementation revision number:

0x2 ETM revision for r1p1

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR1 can be accessed through the external debug interface, offset 0x1E4.

### 5.6.31 TRCIDR2, ID Register 2

The TRCIDR2 returns the maximum size of six parameters in the trace unit.

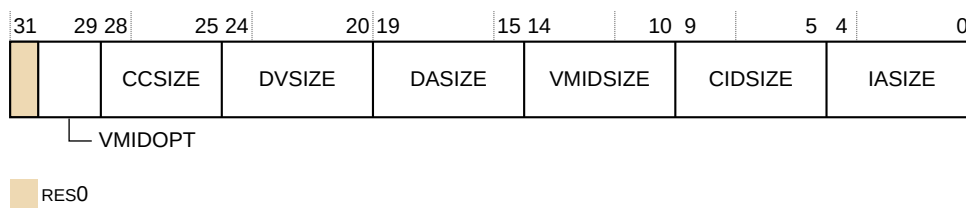
The parameters are:

- Cycle counter.
- Data value.
- Data address.
- VMID.
- Context ID.
- Instruction address.

#### Bit field descriptions

The TRCIDR2 is a 32-bit register.

**Figure 5-62: TRCIDR2 bit assignments**



#### RES0, [31]

*res0*

Reserved.

#### VMIDOPT, [30:29]

Indicates the options for observing the Virtual context identifier:

**0x1**

VMIDOPT is implemented.

#### CCSIZE, [28:25]

Size of the cycle counter in bits minus 12:

**0x0**

The cycle counter is 12 bits in length.

**DVSIZE, [24:20]**

Data value size in bytes:

**0x00**

Data value tracing is not implemented.

**DASIZE, [19:15]**

Data address size in bytes:

**0x00**

Data address tracing is not implemented.

**VMIDSIZE, [14:10]**

Virtual Machine ID size:

**0x4**

Maximum of 32-bit Virtual Machine ID size.

**CIDSIZE, [9:5]**

Context ID size in bytes:

**0x4**

Maximum of 32-bit Context ID size.

**IASIZE, [4:0]**

Instruction address size in bytes:

**0x8**

Maximum of 64-bit address size.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR2 can be accessed through the external debug interface, offset 0x1E8.

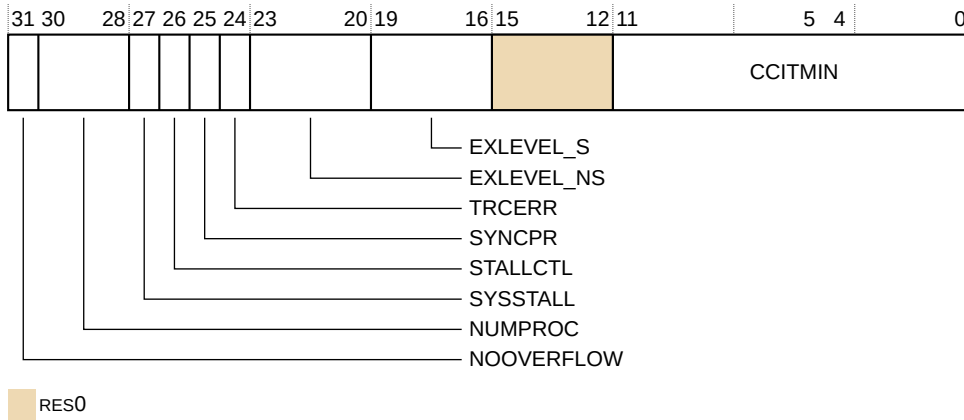
### 5.6.32 TRCIDR3, ID Register 3

The TRCIDR3 indicates:

- Whether TRCVICTLR is supported.
- The number of cores available for tracing.
- If an Exception level supports instruction tracing.
- The minimum threshold value for instruction trace cycle counting.
- Whether the synchronization period is fixed.

#### Bit field descriptions

The TRCIDR3 is a 32-bit register.

**Figure 5-63: TRCIDR3 bit assignments****NOOVERFLOW, [31]**

Indicates whether TRCSTALLCTLR.NOOVERFLOW is implemented:

0 TRCSTALLCTLR.NOOVERFLOW is not implemented.

**NUMPROC, [30:28]**

Indicates the number of cores available for tracing:

0b000 The trace unit can trace one core, ETM trace unit sharing not supported.

**SYSSTALL, [27]**

Indicates whether stall control is implemented:

1 The system supports core stall control.

**STALLCTL, [26]**

Indicates whether TRCSTALLCTLR is implemented:

1 TRCSTALLCTLR is implemented.

This field is used in conjunction with SYSSTALL.

**SYNCPR, [25]**

Indicates whether there is a fixed synchronization period:

0 TRCSYNCPR is read/write so software can change the synchronization period.

**TRCERR, [24]**

Indicates whether TRCVICTLR.TRCERR is implemented:

1 TRCVICTLR.TRCERR is implemented.



EXLEVEL\_NS, [23:20]

Each bit controls whether instruction tracing in Non-secure state is implemented for the corresponding Exception level:

0b0111	Instruction tracing is implemented for Non-secure EL0, EL1, and EL2 Exception levels.
--------	---

EXLEVEL\_S, [19:16]

Each bit controls whether instruction tracing in Secure state is implemented for the corresponding Exception level:

0b1011	Instruction tracing is implemented for Secure EL0, EL1, and EL3 Exception levels.
--------	---

## RES0, [15:12]

<i>res0</i>	Reserved.
-------------	-----------

### CCITMIN, [11:0]

The minimum value that can be programmed in TRCCCCTLR.THRESHOLD:

0x004 Instruction trace cycle counting minimum threshold is 4.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR3 can be accessed through the external debug interface, offset 0x1EC.

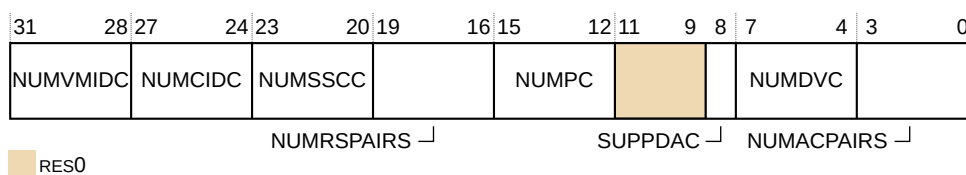
### 5.6.33 TRCIDR4, ID Register 4

The TRCIDR4 indicates the resources available in the ETM trace unit.

## Bit field descriptions

The TRCIDR4 is a 32-bit register.

**Figure 5-64: TRCIDR4 bit assignments**



NUMVMIDC, [31:28]

Indicates the number of VMID comparators available for tracing:

0x1          One VMID comparator is available.

#### **NUMCIDC, [27:24]**

Indicates the number of CID comparators available for tracing:

0x1          One Context ID comparator is available.

#### **NUMSSCC, [23:20]**

Indicates the number of single-shot comparator controls available for tracing:

0x1          One single-shot comparator control is available.

#### **NUMRSPAIRS, [19:16]**

Indicates the number of resource selection pairs available for tracing:

0x7          Eight resource selection pairs are available.

#### **NUMPC, [15:12]**

Indicates the number of core comparator inputs available for tracing:

0x0          Core comparator inputs are not implemented.

#### **RES0, [11:9]**

RES0          Reserved.

#### **SUPPDAC, [8]**

Indicates whether the implementation supports data address comparisons: This value is:

0            Data address comparisons are not implemented.

#### **NUMDVC, [7:4]**

Indicates the number of data value comparators available for tracing:

0x0          Data value comparators are not implemented.

#### **NUMACPAIRS, [3:0]**

Indicates the number of address comparator pairs available for tracing:

0x4          Four address comparator pairs are implemented.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

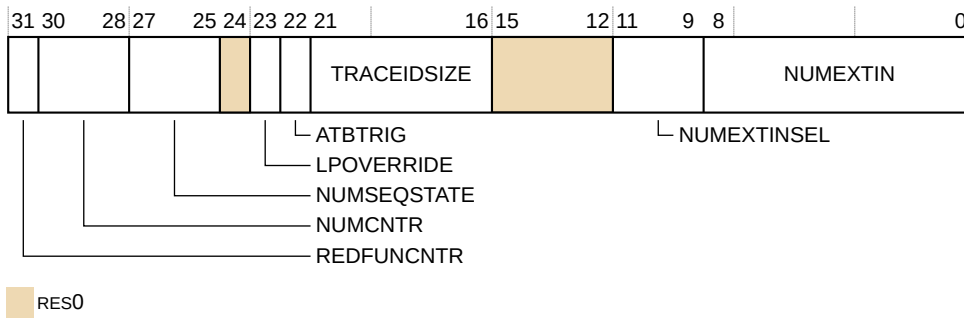
The TRCIDR4 can be accessed through the external debug interface, offset 0x1F0.

### 5.6.34 TRCIDR5, ID Register 5

The TRCIDR5 returns how many resources the trace unit supports.

#### Bit field descriptions

**Figure 5-65: TRCIDR5 bit assignments**



#### REDFUNCNTR, [31]

Reduced Function Counter implemented:

0 Reduced Function Counter not implemented.

#### NUMCNTR, [30:28]

Number of counters implemented:

0b010 Two counters implemented.

#### NUMSEQSTATE, [27:25]

Number of sequencer states implemented:

0b100 Four sequencer states implemented.

#### RES0, [24]

RES0 Reserved.

#### LPOVERRIDE, [23]

Low-power state override support:

1 Low-power state override support is implemented.

#### ATBTRIG, [22]

ATB trigger support:

1 ATB trigger support implemented.

**TRACEIDSIZE, [21:16]**

Number of bits of trace ID:

0x07      Seven-bit trace ID implemented.

**RES0, [15:12]**

RES0      Reserved.

**NUMEXTINSEL, [11:9]**

Number of external input selectors implemented:

0b100      Four external input selectors implemented.

**NUMEXTIN, [8:0]**

Number of external inputs implemented:

0x2F      47 external inputs implemented.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

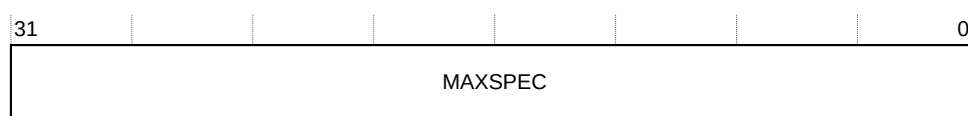
The TRCIDR5 can be accessed through the external debug interface, offset 0x1F4.

### 5.6.35 TRCIDR8, ID Register 8

The TRCIDR8 returns the maximum speculation depth of the instruction trace stream.

**Bit field descriptions**

The TRCIDR8 is a 32-bit register.

**Figure 5-66: TRCIDR8 bit assignments****MAXSPEC, [31:0]**

The maximum number of PO elements in the trace stream that can be Speculative at any time.

0      Maximum speculation depth of the instruction trace stream.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR8 can be accessed through the external debug interface, offset 0x180.

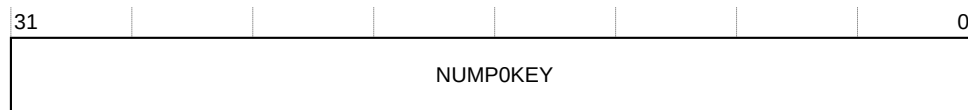
### 5.6.36 TRCIDR9, ID Register 9

The TRCIDR9 returns the number of P0 right-hand keys that the trace unit can use.

#### Bit field descriptions

The TRCIDR9 is a 32-bit register.

**Figure 5-67: TRCIDR9 bit assignments**



#### NUMPOKEY, [31:0]

The number of P0 right-hand keys that the trace unit can use.

0                      Number of P0 right-hand keys.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR9 can be accessed through the external debug interface, offset 0x184.

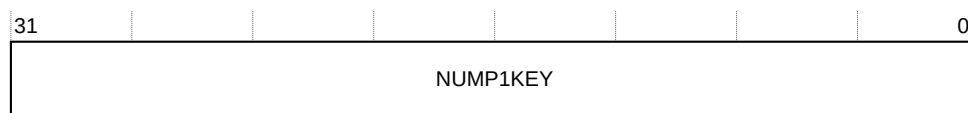
### 5.6.37 TRCIDR10, ID Register 10

The TRCIDR10 returns the number of P1 right-hand keys that the trace unit can use.

#### Bit field descriptions

The TRCIDR10 is a 32-bit register.

**Figure 5-68: TRCIDR10 bit assignments**



#### NUMP1KEY, [31:0]

The number of P1 right-hand keys that the trace unit can use.

0                      Number of P1 right-hand keys.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR10 can be accessed through the external debug interface, offset 0x188.

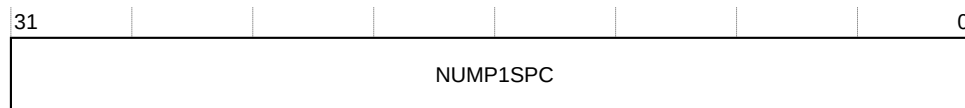
### 5.6.38 TRCIDR11, ID Register 11

The TRCIDR11 returns the number of special P1 right-hand keys that the trace unit can use.

#### Bit field descriptions

The TRCIDR11 is a 32-bit register.

**Figure 5-69: TRCIDR11 bit assignments**



#### NUMP1SPC, [31:0]

The number of special P1 right-hand keys that the trace unit can use.

0                      Number of special P1 right-hand keys.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR11 can be accessed through the external debug interface, offset 0x18C.

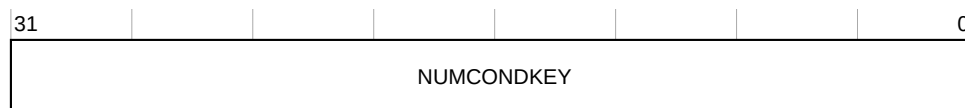
### 5.6.39 TRCIDR12, ID Register 12

The TRCIDR12 returns the number of conditional instruction right-hand keys that the trace unit can use.

#### Bit field descriptions

The TRCIDR12 is a 32-bit register.

**Figure 5-70: TRCIDR12 bit assignments**



#### NUMCONDKEY, [31:0]

The number of conditional instruction right-hand keys that the trace unit can use, including normal and special keys.

0                      Number of conditional instruction right-hand keys.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR12 can be accessed through the external debug interface, offset 0x190.

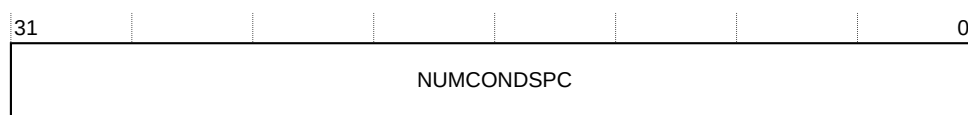
### 5.6.40 TRCIDR13, ID Register 13

The TRCIDR13 returns the number of special conditional instruction right-hand keys that the trace unit can use.

#### Bit field descriptions

The TRCIDR13 is a 32-bit register.

**Figure 5-71: TRCIDR13 bit assignments**



#### NUMCONDSPC, [31:0]

The number of special conditional instruction right-hand keys that the trace unit can use, including normal and special keys.

0 Number of special conditional instruction right-hand keys.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR13 can be accessed through the external debug interface, offset 0x194.

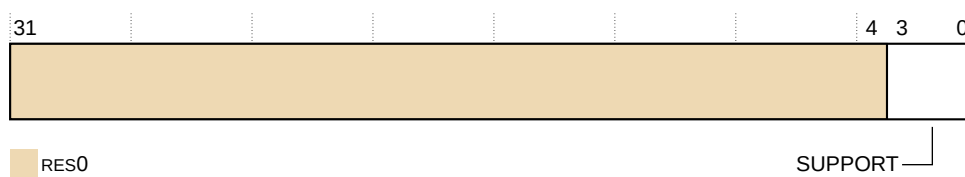
### 5.6.41 TRCIMSPECO, Implementation Specific Register 0

The TRCIMSPECO shows the presence of any *implementation specific* features, and enables any features that are provided.

#### Bit field descriptions

The TRCIMSPECO is a 32-bit register.

**Figure 5-72: TRCIMSPECO bit assignments**



**RES0, [31:4]****res0**

Reserved.

**SUPPORT, [3:0]****0**No *implementation specific* extensions are supported.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIMSPECO can be accessed through the external debug interface, offset 0x1C0.

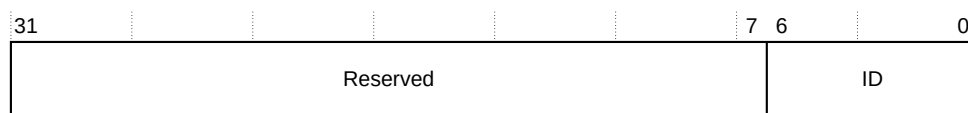
### 5.6.42 TRCITATBIDR, Integration ATB Identification Register

The TRCITATBIDR sets the state of output pins, which are mentioned in the bit descriptions in this section.

**Bit field descriptions**

The TRCITATBIDR is a 32-bit register.

**Figure 5-73: TRCITATBIDR bit assignments**

**[31:7]**

Reserved. Read undefined.

**ID, [6:0]**Drives the **ATIDMn[6:0]** output pins.

When a bit is set to 0, the corresponding output pin is LOW.

When a bit is set to 1, the corresponding output pin is HIGH.

The TRCITATBIDR bit values correspond to the physical state of the output pins.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITATBIDR can be accessed through the external debug interface, offset 0xEE4.



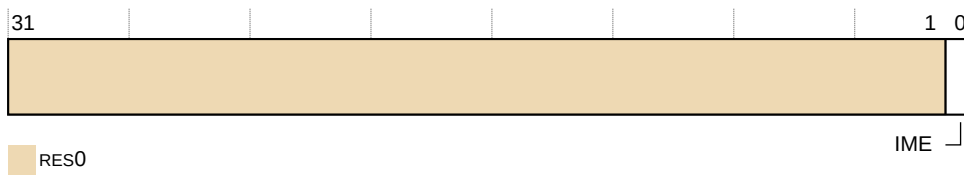
### 5.6.43 TRCITCTRL, Integration Mode Control Register

The TRCITCTRL enables topology detection or integration testing, by putting the ETM trace unit into integration mode.

#### Bit field descriptions

The TRCITCTRL is a 32-bit register.

**Figure 5-74: TRCITCTRL bit assignments**



#### RES0, [31:1]

**RES0** Reserved.

#### IME, [0]

Integration mode enable bit. The possible values are:

- |   |   |
|---|---|
| 0 | The trace unit is not in integration mode.  |
| 1 | The trace unit is in integration mode. This mode enables: <ul style="list-style-type: none"> <li>• A debug agent to perform topology detection.</li> <li>• SoC test software to perform integration testing.</li> </ul> |

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITCTRL can be accessed through the external debug interface, offset 0xF00.

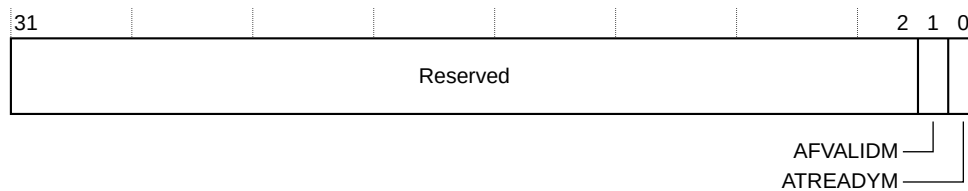
### 5.6.44 TRCITIATBINR, Integration Instruction ATB In Register

The TRCITIATBINR reads the state of the input pins that are described in this section.

#### Bit field descriptions

The TRCITIATBINR is a 32-bit register.

### Figure 5-75: TRCITIATBINR bit assignments



For all non-reserved bits:

- When an input pin is LOW, the corresponding register bit is 0.
- When an input pin is HIGH, the corresponding register bit is 1.
- The TRCITIATBINR bit values always correspond to the physical state of the input pins.

[31:2]

Reserved. Read undefined.

AFVALIDM, [1]

Returns the value of the **AFVALIDMn** input pin.

ATREADYM, [0]

Returns the value of the **ATREADYMn** input pin.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITIATBINR can be accessed through the external debug interface, offset 0xEF4.

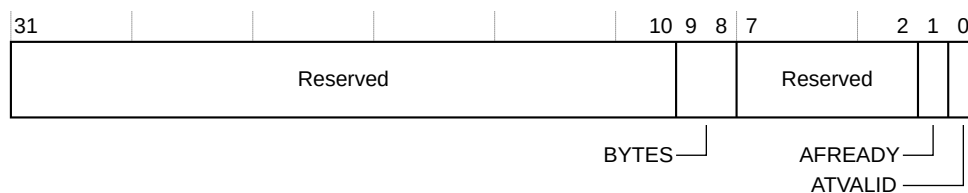
### 5.6.45 TRCITIATBOUTR, Integration Instruction ATB Out Register

The TRCITIATBOUTR sets the state of the output pins that are mentioned in the bit descriptions in this section.

## Bit field descriptions

The TRCITIATBOUTR is a 32-bit register.

### Figure 5-76: TRCITIATBOUTR bit assignments



For all non-reserved bits:

- When a bit is set to 0, the corresponding output pin is LOW.

- When a bit is set to 1, the corresponding output pin is HIGH.
- The TRCITIATBOUTR bit values always correspond to the physical state of the output pins.

**[31:10]**

Reserved. Read undefined.

**BYTES, [9:8]**Drives the **ATBYTESMn[1:0]** output pins.**[7:2]**

Reserved. Read undefined.

**AFREADY, [1]**Drives the **AFREADYMn** output pin.**ATVALID, [0]**Drives the **ATVALIDMn** output pin.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITIATBOUTR can be accessed through the external debug interface, offset 0xEFC.

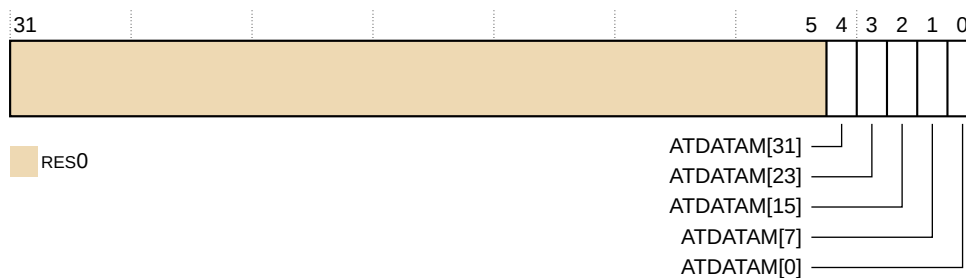
### 5.6.46 TRCITIDATAR, Integration Instruction ATB Data Register

The TRCITIDATAR sets the state of the **ATDATAMn** output pins that are shown in the TRCITIDATAR bit assignments table.

#### Bit field descriptions

The TRCITIDATAR is a 32-bit register.

**Figure 5-77: TRCITIDATAR bit assignments**

**RES0, [31:5]**

**RES0** Reserved.

## ATDATAM[31], [4]

Drives the ATDATAM[31] output. <sup>3</sup>

## ATDATAM[23], [3]

Drives the ATDATAM[23] output.<sup>3</sup>

## ATDATAM[15], [2]

Drives the ATDATAM[15] output.<sup>3</sup>

## ATDATAM[7], [1]

Drives the ATDATAM[7] output.<sup>3</sup>

ATDATAM[0], [0]

Drives the ATDATAM[0] output.<sup>3</sup>

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITIDATAR can be accessed through the external debug interface, offset 0xEEC.

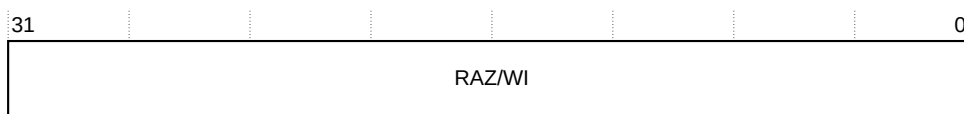
### 5.6.47 TRCLAR, Software Lock Access Register

The TRCLAR controls access to registers using the memory-mapped interface, when **PADDRDBG31** is LOW.

## Bit field descriptions

The TRCLAR is a 32-bit register.

**Figure 5-78: TRCLAR bit assignments**



## RAZ/WI, [31:0]

Read-As-Zero, write ignore.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCLAR can be accessed through the external debug interface, offset 0xFB0.

<sup>3</sup> When a bit is set to 0, the corresponding output pin is LOW. When a bit is set to 1, the corresponding output pin is HIGH. The TRCITIDATAR bit values correspond to the physical state of the output pins.

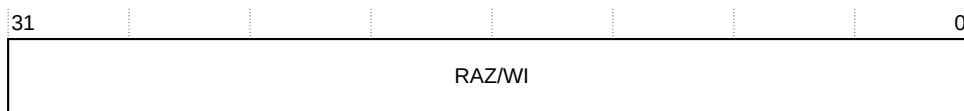
### 5.6.48 TRCLSR, Software Lock Status Register

The TRCLSR determines whether the software lock is implemented, and indicates the current status of the software lock.

#### Bit field descriptions

The TRCLSR is a 32-bit register.

**Figure 5-79: TRCLSR bit assignments**



#### RAZ/WI, [31:0]

Read-As-Zero, write ignore.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCLSR can be accessed through the external debug interface, offset 0xFB4.

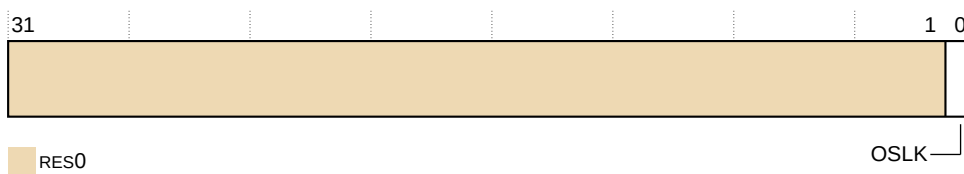
### 5.6.49 TRCOSLAR, OS Lock Access Register

The TRCOSLAR sets and clears the OS Lock, to lock out external debugger accesses to the ETM trace unit registers.

#### Bit field descriptions

The TRCOSLAR is a 32-bit register.

**Figure 5-80: TRCOSLAR bit assignments**



#### RES0, [31:1]

RES0 Reserved.

#### OSLK, [0]

OS Lock key value:

0	Unlock the OS Lock.
1	Lock the OS Lock.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCOSLAR can be accessed through the external debug interface, offset 0x300.

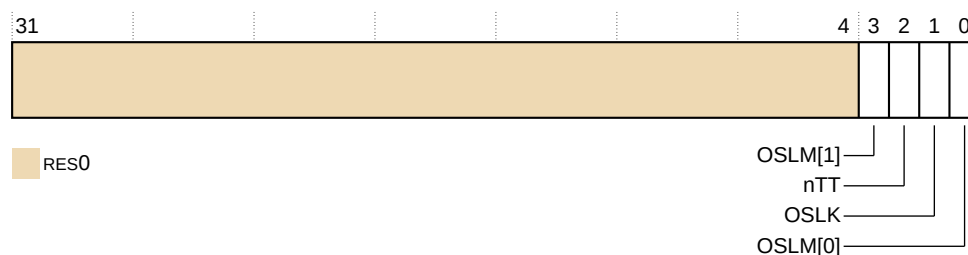
### 5.6.50 TRCOSLSR, OS Lock Status Register

The TRCOSLSR returns the status of the OS Lock.

#### Bit field descriptions

The TRCOSLSR is a 32-bit register.

**Figure 5-81: TRCOSLSR bit assignments**



#### RES0, [31:4]

RES0	Reserved.
------	-----------

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

OS Lock model [1] bit. This bit is combined with OSLM[0] to form a two-bit field that indicates the OS Lock model is implemented.

The value of this field is always 0b10, indicating that the OS Lock is implemented.

#### nTT, [2]

This bit is RAZ, that indicates that software must perform a 32-bit write to update the TRCOSLAR.

#### OSLK, [1]

OS Lock status bit:

0	OS Lock is unlocked.
1	OS Lock is locked.

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

OS Lock model [0] bit. This bit is combined with OSLM[1] to form a two-bit field that indicates the OS Lock model is implemented.

The value of this field is always 0b10, indicating that the OS Lock is implemented.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCOSLSR can be accessed through the external debug interface, offset 0x304.

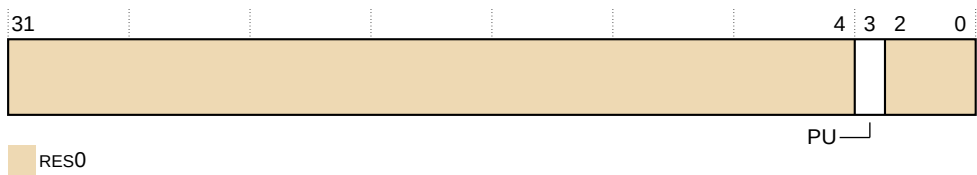
**5.6.51 TRCPDCR, Power Down Control Register**

The TRCPDCR request to the system power controller to keep the ETM trace unit powered up.

**Bit field descriptions**

The TRCPDCR is a 32-bit register.

**Figure 5-82: TRCPDCR bit assignments**



**RES0, [31:4]**

**RES0**      Reserved.

**PU, [3]**

Powerup request, to request that power to the ETM trace unit and access to the trace registers is maintained:

0	Power not requested.
1	Power requested.

This bit is reset to 0 on a trace unit reset.

**RES0, [2:0]**

**RES0**      Reserved.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPDCR can be accessed through the external debug interface, offset 0x310.

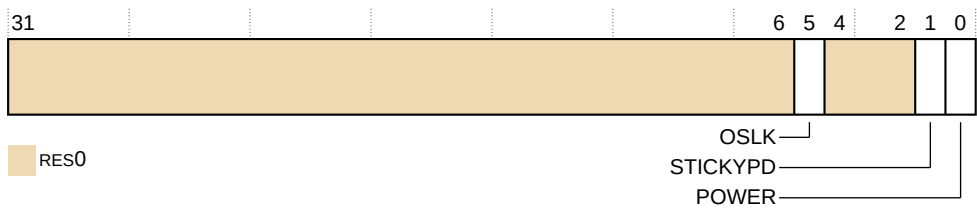
### 5.6.52 TRCPDSR, Power Down Status Register

The TRCPDSR indicates the power down status of the ETM trace unit.

#### Bit field descriptions

The TRCPDSR is a 32-bit register.

Figure 5-83: TRCPDSR bit assignments



#### RES0, [31:6]

RES0      Reserved.

#### OSLK, [5]

OS lock status.

- |   |                          |
|---|--------------------------|
| 0 | The OS Lock is unlocked. |
| 1 | The OS Lock is locked.   |

#### RES0, [4:2]

RES0      Reserved.

#### STICKYPD, [1]

Sticky power down state.

- |   |  |
|---|--|
| 0 | Trace register power has not been removed since the TRCPDSR was last read. |
| 1 | Trace register power has been removed since the TRCPDSR was last read.     |

This bit is set to 1 when power to the ETM trace unit registers is removed, to indicate that programming state has been lost. It is cleared after a read of the TRCPDSR.

#### POWER, [0]

Indicates the ETM trace unit is powered:

- |   |  |
|---|--|
| 0 | ETM trace unit is not powered. The trace registers are not accessible and they all return an error response. |
| 1 | ETM trace unit is powered. All registers are accessible.   |



If a system implementation allows the ETM trace unit to be powered off independently of the Debug power domain, the system must handle accesses to the ETM trace unit appropriately.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPDSR can be accessed through the external debug interface, offset 0x314.

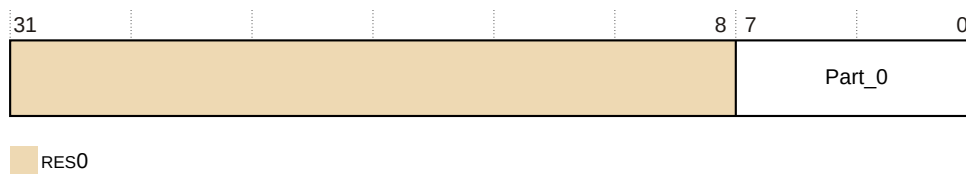
### 5.6.53 TRCPIDR0, ETM Peripheral Identification Register 0

The TRCPIDR0 provides information to identify a trace component.

#### Bit field descriptions

The TRCPIDR0 is a 32-bit register.

**Figure 5-84: TRCPIDR0 bit assignments**



#### RES0, [31:8]

**RES0**      Reserved.

#### Part\_0, [7:0]

**0x43**      Least significant byte of the ETM trace unit part number.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

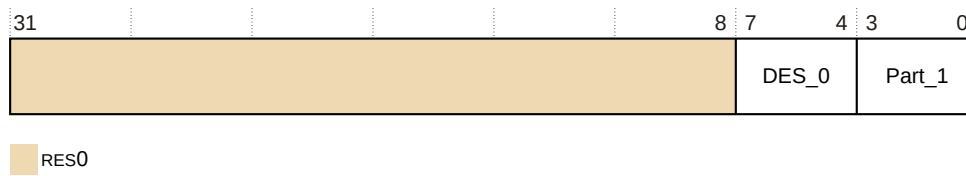
The TRCPIDR0 can be accessed through the external debug interface, offset 0xFE0.

### 5.6.54 TRCPIDR1, ETM Peripheral Identification Register 1

The TRCPIDR1 provides information to identify a trace component.

#### Bit field descriptions

The TRCPIDR1 is a 32-bit register.

**Figure 5-85: TRCPIDR1 bit assignments****RES0, [31:8]**

**RES0** Reserved.

**DES\_0, [7:4]**

**DES\_0** 0xB Arm Limited. This is bits[3:0] of JEP106 ID code.

**Part\_1, [3:0]**

**Part\_1** 0xD Most significant four bits of the ETM trace unit part number.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

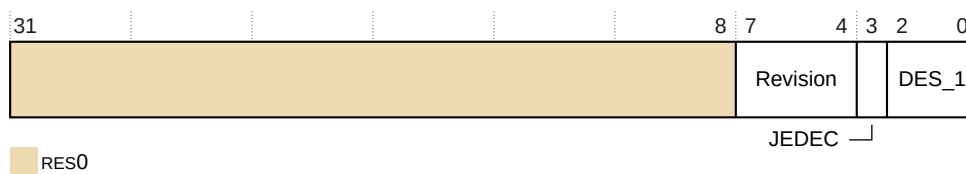
The TRCPIDR1 can be accessed through the external debug interface, offset 0xFE4.

### 5.6.55 TRCPIDR2, ETM Peripheral Identification Register 2

The TRCPIDR2 provides information to identify a trace component.

#### Bit field descriptions

The TRCPIDR2 is a 32-bit register.

**Figure 5-86: TRCPIDR2 bit assignments****RES0, [31:8]**

**RES0** Reserved.

Revision, [7:4]

0x2      r1p1.

JEDEC, [3]

0b1      **RES1.** Indicates a JEP106 identity code is used.

DES\_1, [2:0]

0b011      Arm Limited. This is bits[6:4] of JEP106 ID code.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPIDR2 can be accessed through the external debug interface, offset 0xFE8.

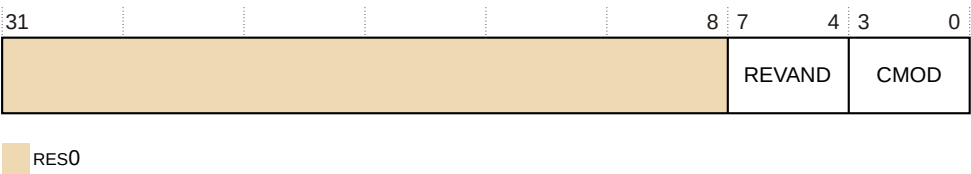
5.6.56 TRCPIDR3, ETM Peripheral Identification Register 3

The TRCPIDR3 provides information to identify a trace component.

Bit field descriptions

The TRCPIDR3 is a 32-bit register.

Figure 5-87: TRCPIDR3 bit assignments



RES0, [31:8]

RES0      Reserved.

REVAND, [7:4]

0x0      Part minor revision.

CMOD, [3:0]

0x0      Not customer modified.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPIDR3 can be accessed through the external debug interface, offset 0xFEC.

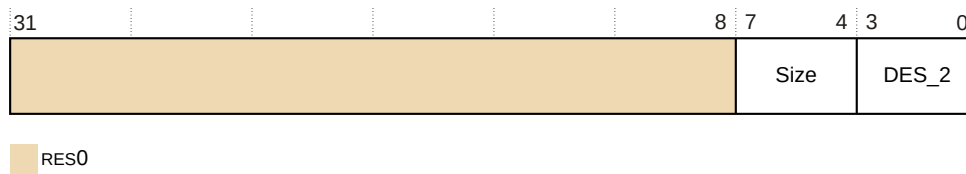
### 5.6.57 TRCPIDR4, ETM Peripheral Identification Register 4

The TRCPIDR4 provides information to identify a trace component.

#### Bit field descriptions

The TRCPIDR4 is a 32-bit register.

**Figure 5-88: TRCPIDR4 bit assignments**



#### RES0, [31:8]

**RES0**      Reserved.

#### Size, [7:4]

**0x0**      Size of the component. Log2 the number of 4KB pages from the start of the component to the end of the component ID registers.

#### DES\_2, [3:0]

**0x4**      Arm Limited. This is bits[3:0] of the JEP106 continuation code.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPIDR4 can be accessed through the external debug interface, offset 0xFD0.

### 5.6.58 TRCPIDRn, ETM Peripheral Identification Registers 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers.

They are reserved for future use and are **RES0**.

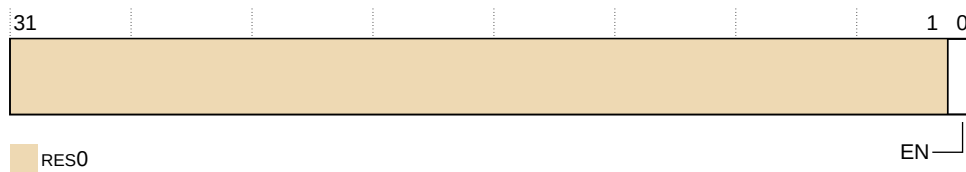
### 5.6.59 TRCPRGCTLR, Programming Control Register

The TRCPRGCTLR enables the ETM trace unit.

#### Bit field descriptions

The TRCPRGCTLR is a 32-bit register.

**Figure 5-89: TRCPRGCTLR bit assignments**



#### RES0, [31:1]

**RES0** Reserved.

#### EN, [0]

Trace program enable:

- |   |  |
|---|--|
| 0 | The ETM trace unit interface in the core is disabled, and clocks are enabled only when necessary to process APB accesses, or drain any already generated trace. This is the reset value. |
| 1 | The ETM trace unit interface in the core is enabled, and clocks are enabled. Writes to most trace registers are <b>IGNORED</b> .   |

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

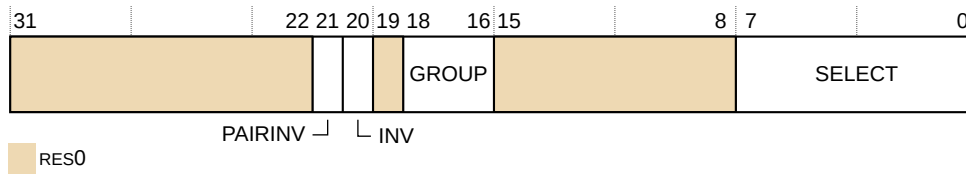
The TRCPRGCTLR can be accessed through the external debug interface, offset 0x004.

### 5.6.60 TRCRSCTLRn, Resource Selection Control Registers 2-15

The TRCRSCTLRn registers control the trace resources. There are eight resource pairs, the first pair is predefined as {0,1,pair=0} and having reserved select registers. This leaves seven pairs to be implemented as programmable selectors.

#### Bit field descriptions

The TRCRSCTLRn registers are 32-bit registers.

**Figure 5-90: TRCRSCTLRn bit assignments****RES0, [31:22]**

**RES0**      Reserved

**PAIRINV, [21]**

Inverts the result of a combined pair of resources.

This bit is implemented only on the lower register for a pair of resource selectors.

**INV, [20]**

Inverts the selected resources:

0	Resource is not inverted.
1	Resource is inverted.

**RES0, [19]**

**RES0**      Reserved

**GROUP, [18:16]**

Selects a group of resources. See the *Arm® ETM Architecture Specification, ETMv4* for more information.

**RES0, [15:8]**

**RES0**      Reserved

**SELECT, [7:0]**

Selects one or more resources from the required group. One bit is provided for each resource from the group.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCRSCTLRn registers can be accessed through the external debug interface, offset 0x208–0x023C.

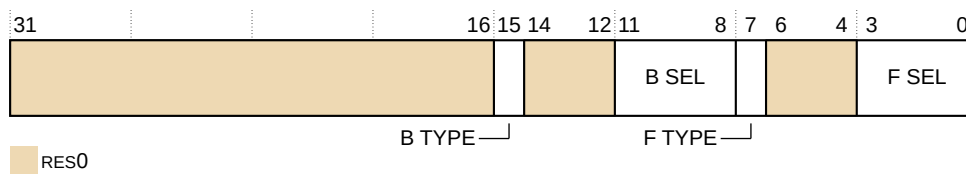
### 5.6.61 TRCSEQEVRn, Sequencer State Transition Control Registers 0-2

The TRCSEQEVRn registers define the sequencer transitions that progress to the next state or backwards to the previous state. The ETM trace unit implements a sequencer state machine with up to four states.

#### Bit field descriptions

The TRCSEQEVRn registers are 32-bit registers.

**Figure 5-91: TRCSEQEVRn bit assignments**



#### RES0, [31:16]

RES0      Reserved

#### B TYPE, [15]

Selects the resource type to move backwards to this state from the next state:

0	Single selected resource
1	Boolean combined resource pair

#### RES0, [14:12]

RES0      Reserved

#### B SEL, [11:8]

Selects the resource number, based on the value of B TYPE:

When B TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When B TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

#### F TYPE, [7]

Selects the resource type to move forwards from this state to the next state:

0	Single selected resource
1	Boolean combined resource pair

#### RES0, [6:4]

RES0      Reserved

**F SEL, [3:0]**

Selects the resource number, based on the value of F TYPE:

When F TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When F TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSEQEVRn registers can be accessed through the external debug interface, offsets:

**TRCSEQEVR0**

0x100

**TRCSEQEVR1**

0x104

**TRCSEQEVR2**

0x108

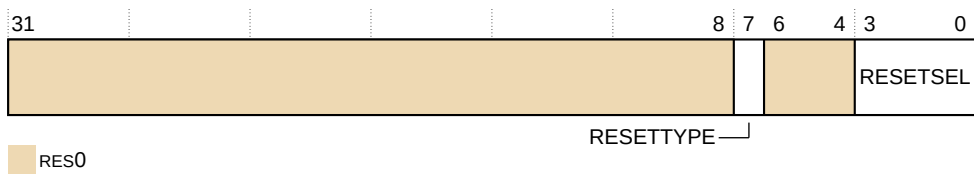
**5.6.62 TRCSEQRSTEV, Sequencer Reset Control Register**

The TRCSEQRSTEV resets the sequencer to state 0.

**Bit field descriptions**

The TRCSEQRSTEV is a 32-bit register.

**Figure 5-92: TRCSEQRSTEV bit assignments**

**RES0, [31:8]**

**RES0** Reserved.

**RESETTYPE, [7]**

Selects the resource type to move back to state 0:

0	Single selected resource.
1	Boolean combined resource pair.



**RES0, [6:4]**

**RES0** Reserved.

**RESETSEL, [3:0]**

Selects the resource number, based on the value of RESETTYPE:

When RESETTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When RESETTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSEQRSTEVSR can be accessed through the external debug interface, offset 0x118.

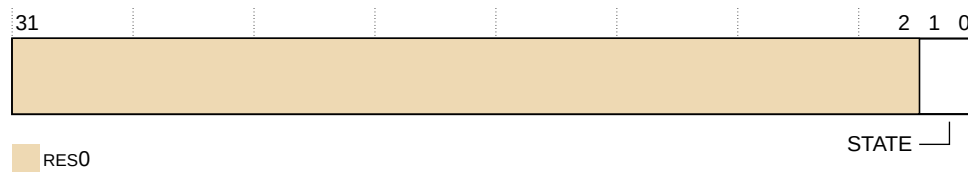
**5.6.63 TRCSEQSTR, Sequencer State Register**

The TRCSEQSTR holds the value of the current state of the sequencer.

**Bit field descriptions**

The TRCSEQSTR is a 32-bit register.

**Figure 5-93: TRCSEQSTR bit assignments**

**RES0, [31:2]**

**RES0** Reserved.

**STATE, [1:0]**

Current sequencer state:

0b00	State 0.
0b01	State 1.
0b10	State 2.
0b11	State 3.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSEQSTR can be accessed through the external debug interface, offset 0x11C.

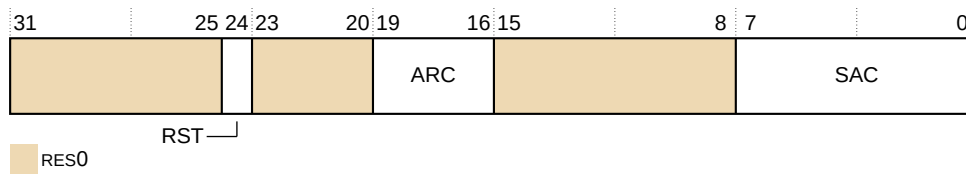
### 5.6.64 TRCSSCCR0, Single-Shot Comparator Control Register 0

The TRCSSCCR0 controls the single-shot comparator.

#### Bit field descriptions

The TRCSSCCR0 is a 32-bit register.

**Figure 5-94: TRCSSCCR0 bit assignments**



#### RES0, [31:25]

**RES0** Reserved.

#### RST, [24]

Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected:

1 Reset enabled. Multiple matches can occur.

#### RES0, [23:20]

**RES0** Reserved.

#### ARC, [19:16]

Selects one or more address range comparators for single-shot control.

One bit is provided for each implemented address range comparator.

#### RES0, [15:8]

**RES0** Reserved.

#### SAC, [7:0]

Selects one or more single address comparators for single-shot control.

One bit is provided for each implemented single address comparator.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSSCCR0 can be accessed through the external debug interface, offset 0x280.

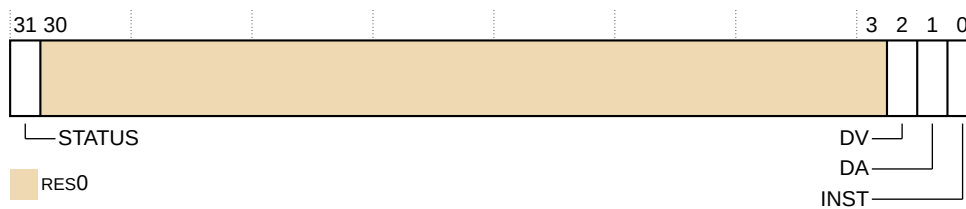
### 5.6.65 TRCSSCSR0, Single-Shot Comparator Status Register 0

The TRCSSCSR0 indicates the status of the single-shot comparator. TRCSSCSR0 is sensitive to instruction addresses.

#### Bit field descriptions

The TRCSSCSR0 is a 32-bit register.

**Figure 5-95: TRCSSCSR0 bit assignments**



#### STATUS, [31]

Single-shot status. This indicates whether any of the selected comparators have matched:

- |   |                                   |
|---|-----------------------------------|
| 0 | Match has not occurred.           |
| 1 | Match has occurred at least once. |

When programming the ETM trace unit, if TRCSSCCRn.RST is b0, the STATUS bit must be explicitly written to 0 to enable this single-shot comparator control.

#### RES0, [30:3]

- |      |           |
|------|-----------|
| RES0 | Reserved. |
|------|-----------|

#### DV, [2]

Data value comparator support:

- |   |   |
|---|---|
| 0 | Single-shot data value comparisons not supported. |
|---|---|

#### DA, [1]

Data address comparator support:

- |   |   |
|---|---|
| 0 | Single-shot data address comparisons not supported. |
|---|---|

#### INST, [0]

Instruction address comparator support:

- |   |  |
|---|--|
| 1 | Single-shot instruction address comparisons supported. |
|---|--|

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSSCSR0 can be accessed through the external debug interface, offset 0x2A0.

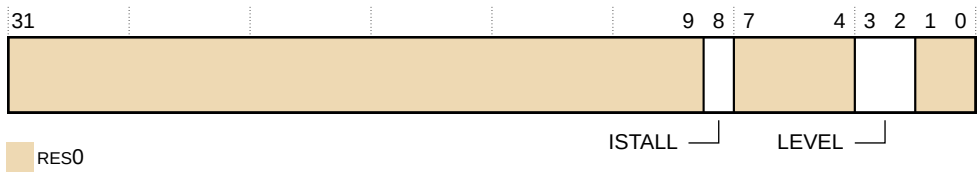
### 5.6.66 TRCSTALLCTLR, Stall Control Register

The TRCSTALLCTLR enables the ETM trace unit to stall the Cortex®-A65AE core if the ETM trace unit FIFO overflows.

#### Bit field descriptions

The TRCSTALLCTLR is a 32-bit register.

Figure 5-96: TRCSTALLCTLR bit assignments



#### RES0, [31:9]

RES0      Reserved.

#### ISTALL, [8]

Instruction stall bit. Controls if the trace unit can stall the core when the instruction trace buffer space is less than LEVEL:

- |   |   |
|---|---|
| 0 | The trace unit does not stall the core. |
| 1 | The trace unit can stall the core.      |

#### RES0, [7:4]

RES0      Reserved.

#### LEVEL, [3:2]

Threshold level field. The field can support 4 monotonic levels from 0b00 to 0b11, where:

- |      |  |
|------|--|
| 0b00 | Zero invasion. This setting has a greater risk of an ETM trace unit FIFO overflow. |
| 0b11 | Maximum invasion occurs but there is less risk of a FIFO overflow.                 |

#### RES0, [1:0]

RES0      Reserved.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSTALLCTLR can be accessed through the external debug interface, offset 0x02C.

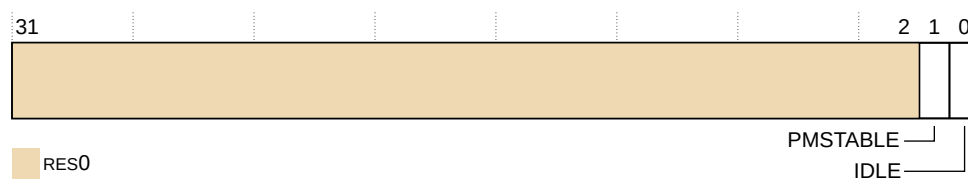
### 5.6.67 TRCSTATR, Status Register

The TRCSTATR indicates the ETM trace unit status.

#### Bit field descriptions

The TRCSTATR is a 32-bit register.

**Figure 5-97: TRCSTATR bit assignments**



#### RES0, [31:2]

**RES0** Reserved.

#### PMSTABLE, [1]

Indicates whether the ETM trace unit registers are stable and can be read:

0	The programmers model is not stable.
1	The programmers model is stable.

#### IDLE, [0]

Idle status:

0	The ETM trace unit is not idle.
1	The ETM trace unit is idle.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSTATR can be accessed through the external debug interface, offset 0x00C.

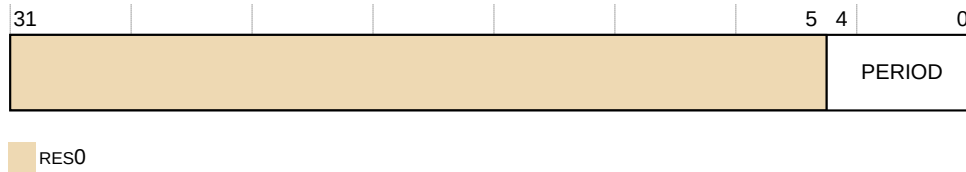
### 5.6.68 TRCSYNCP, Synchronization Period Register

The TRCSYNCP controls how often periodic trace synchronization requests occur.

#### Bit field descriptions

The TRCSYNCP is a 32-bit register.

**Figure 5-98: TRCSYNCP bit assignments**



#### RES0, [31:5]

RES0 Reserved.

#### PERIOD, [4:0]

Defines the number of bytes of trace between synchronization requests as a total of the number of bytes generated by both the instruction and data streams. The number of bytes is  $2^N$  where N is the value of this field:

- A value of zero disables these periodic synchronization requests, but does not disable other synchronization requests.
- The minimum value that can be programmed, other than zero, is 8, providing a minimum synchronization period of 256 bytes.
- The maximum value is 20, providing a maximum synchronization period of  $2^{20}$  bytes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

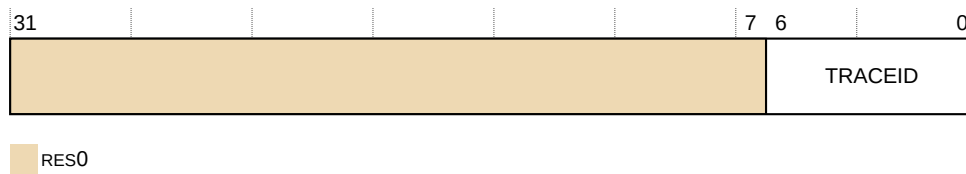
The TRCSYNCP can be accessed through the external debug interface, offset 0x034.

### 5.6.69 TRCTRACEIDR, Trace ID Register

The TRCTRACEIDR sets the trace ID for instruction trace.

#### Bit field descriptions

The TRCTRACEIDR is a 32-bit register.

**Figure 5-99: TRCTRACEIDR bit Assignments****RES0, [31:7]**

**RES0** Reserved.

**TRACEID, [6:0]**

Trace ID value. When only instruction tracing is enabled, this provides the trace ID.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

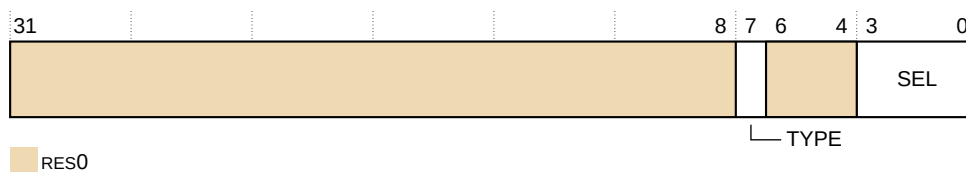
The TRCTRACEIDR can be accessed through the external debug interface, offset 0x040.

**5.6.70 TRCTSCTLR, Global Timestamp Control Register**

The TRCTSCTLR controls the insertion of global timestamps in the trace streams. When the selected event is triggered, the trace unit inserts a global timestamp into the trace streams. The event is selected from one of the Resource Selectors.

**Bit field descriptions**

The TRCTSCTLR is a 32-bit register.

**Figure 5-100: TRCTSCTLR bit assignments****RES0, [31:8]**

**RES0** Reserved

**TYPE, [7]**

Single or combined resource selector.

**RES0, [6:4]**

**RES0**      Reserved

**SEL, [3:0]**

Identifies the resource selector to use.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

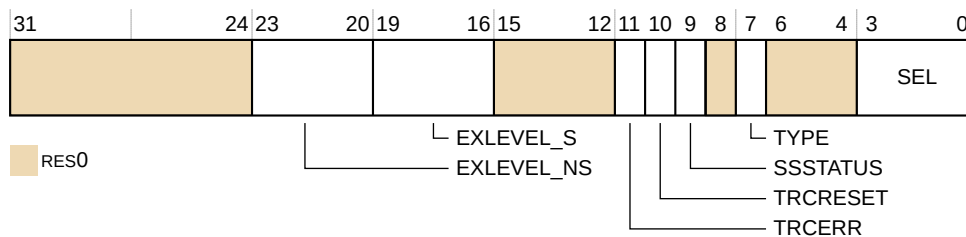
The TRCTSCTLR can be accessed through the external debug interface, offset 0x030.

**5.6.71 TRCVICTLR, ViewInst Main Control Register**

The TRCVICTLR controls instruction trace filtering.

**Bit field descriptions**

The TRCVICTLR is a 32-bit register.

**Figure 5-101: TRCVICTLR bit assignments****RES0, [31:24]**

**RES0**      Reserved.

**EXLEVEL\_NS, [23:20]**

In Non-secure state, each bit controls whether instruction tracing is enabled for the corresponding Exception level:

- |   |   |
|---|---|
| 0 | Trace unit generates instruction trace, in Non-secure state, for Exception level <i>n</i> .         |
| 1 | Trace unit does not generate instruction trace, in Non-secure state, for Exception level <i>n</i> . |

The Exception levels are:

- |         |                    |
|---------|--------------------|
| Bit[20] | Exception level 0. |
| Bit[21] | Exception level 1. |



Bit[22]	Exception level 2.
Bit[23]	RAZ/WI. Instruction tracing is not implemented for Exception level 3.

**EXLEVEL\_S, [19:16]**

In Secure state, each bit controls whether instruction tracing is enabled for the corresponding Exception level:

0	Trace unit generates instruction trace, in Secure state, for Exception level <i>n</i> .
1	Trace unit does not generate instruction trace, in Secure state, for Exception level <i>n</i> .

The Exception levels are:

Bit[16]	Exception level 0.
Bit[17]	Exception level 1.
Bit[18]	RAZ/WI. Instruction tracing is not implemented for Exception level 2.
Bit[19]	Exception level 3.

**RES0, [15:12]**

<b>RES0</b>	Reserved.
-------------	-----------

**TRCERR, [11]**

Selects whether a system error exception must always be traced:

0	System error exception is traced only if the instruction or exception immediately before the system error exception is traced.
1	System error exception is always traced regardless of the value of ViewInst.

**TRCRESET, [10]**

Selects whether a reset exception must always be traced:

0	Reset exception is traced only if the instruction or exception immediately before the reset exception is traced.
1	Reset exception is always traced regardless of the value of ViewInst.

**SSSTATUS, [9]**

Indicates the current status of the start/stop logic:

0	Start/stop logic is in the stopped state.
1	Start/stop logic is in the started state.

**RES0, [8]**

<b>RES0</b>	Reserved.
-------------	-----------

**TYPE, [7]**

Selects the resource type for the viewinst event:

- 0Single selected resource.
- 1Boolean combined resource pair.

RES0, [6:4]

- RES0Reserved.

SEL, [3:0]

Selects the resource number to use for the viewinst event, based on the value of TYPE:

When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCVICTLR can be accessed through the external debug interface, offset 0x080.

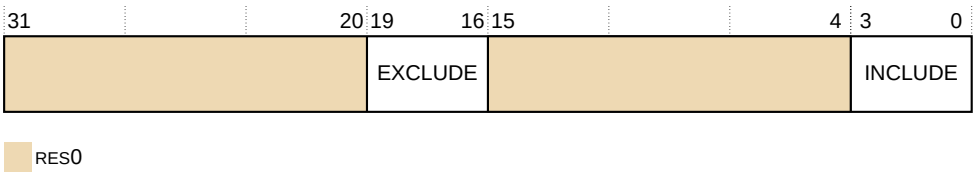
5.6.72 TRCVIIECTLR, ViewInst Include-Exclude Control Register

The TRCVIIECTLR defines the address range comparators that control the ViewInst include/exclude control.

Bit field descriptions

The TRCVIIECTLR is a 32-bit register.

Figure 5-102: TRCVIIECTLR bit assignments



RES0, [31:20]

- RES0Reserved.

EXCLUDE, [19:16]

Defines the address range comparators for ViewInst exclude control. One bit is provided for each implemented Address Range Comparator.

RES0, [15:4]

- RES0Reserved.

**INCLUDE, [3:0]**

Defines the address range comparators for ViewInst include control.

Selecting no include comparators indicates that all instructions must be included. The exclude control indicates which ranges must be excluded.

One bit is provided for each implemented Address Range Comparator.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCVIIECTLR can be accessed through the external debug interface, offset 0x084.

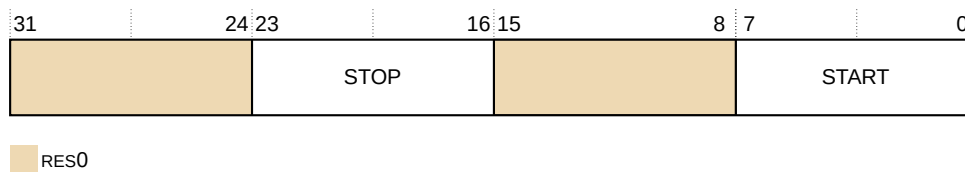
**5.6.73 TRCVISSCTLR, ViewInst Start-Stop Control Register**

The TRCVISSCTLR defines the single address comparators that control the ViewInst Start/Stop logic.

**Bit field descriptions**

The TRCVISSCTLR is a 32-bit register.

**Figure 5-103: TRCVISSCTLR bit assignments**

**RES0, [31:24]**

**RES0** Reserved.

**STOP, [23:16]**

Defines the single address comparators to stop trace with the ViewInst Start/Stop control.

One bit is provided for each implemented single address comparator.

**RES0, [15:8]**

**RES0** Reserved.

**START, [7:0]**

Defines the single address comparators to start trace with the ViewInst Start/Stop control.

One bit is provided for each implemented single address comparator.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCVISSCTLR can be accessed through the external debug interface, offset 0x088.

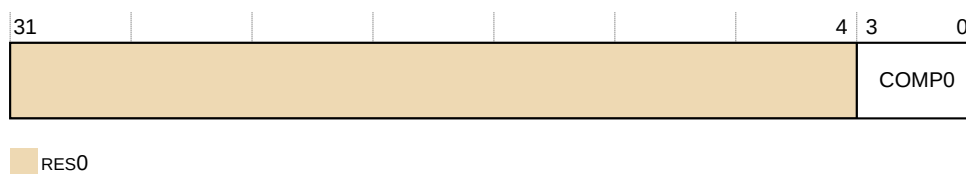
### 5.6.74 TRCVMIDCCTLR0, Virtual context identifier Comparator Control Register 0

The TRCVMIDCCTLR0 contains the Virtual Machine Identifier mask value for the TRCVMIDCVR0 register.

#### Bit field descriptions

The TRCVMIDCCTLR0 is a 32-bit register.

**Figure 5-104: TRCVMIDCCTLR0 bit assignments**



#### RES0, [31:4]

**RES0**      Reserved.

#### COMP0, [3:0]

Controls the mask value that the trace unit applies to TRCVMIDCVR0. Each bit in this field corresponds to a byte in TRCVMIDCVR0. When a bit is:

- |   |  |
|---|--|
| 0 | The trace unit includes the relevant byte in TRCVMIDCVR0 when it performs the Virtual context ID comparison. |
| 1 | The trace unit ignores the relevant byte in TRCVMIDCVR0 when it performs the Virtual context ID comparison.  |

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCVMIDCCTLR0 can be accessed through the external debug interface, offset 0x688.

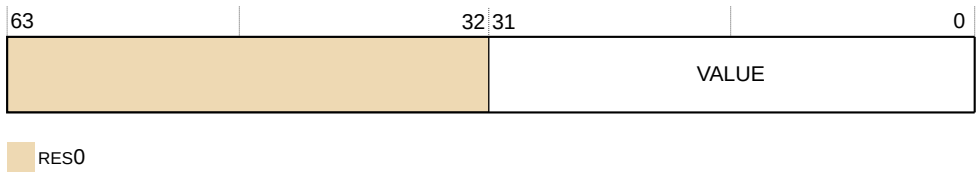
### 5.6.75 TRCVMIDCVR0, VMID Comparator Value Register 0

The TRCVMIDCVR0 contains a VMID value.

#### Bit field descriptions

The TRCVMIDCVR0 is a 64-bit register.

Figure 5-105: TRCVMIDCVR0 bit assignments



#### RES0, [63:32]

RES0      Reserved.

#### VALUE, [31:0]

The VMID value.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCVMIDCVR0 can be accessed through the internal memory-mapped interface and the external debug interface, offset 0x640.

#### Usage constraints

Accepts writes only when the trace unit is disabled.

#### Configurations

Available in all configurations.

# Appendix A Revisions

This appendix describes the technical changes between released issues of this book.

## A.1 Revisions

This appendix describes the technical changes between released issues of this book.

**Table A-1: Issue 0000-00**

Change	Location
First confidential release for r0p0	-

**Table A-2: Differences between Issue 0000-00 and Issue 0100-00**

Change	Location
First confidential release for r1p0	-
Editorial changes	Entire manual
Product name changed from Helios-AE to Cortex-A65AE	Entire manual
Updated the product revision to r1p0	<a href="#">3.1.64 MIDR_EL1, Main ID Register, EL1</a> on page 168  <a href="#">5.2.13 EDPIDR2, External Debug Peripheral Identification Register 2</a> on page 287  <a href="#">5.4.9 PMPIDR2, Performance Monitors Peripheral Identification Register 2</a> on page 309  <a href="#">5.6.55 TRCPIDR2, ETM Peripheral Identification Register 2</a> on page 370
Added information about Hybrid-mode	<a href="#">2.1.1 About the core</a> on page 19
Added information for Lock-mode	<a href="#">2.1.4 Implementation options</a> on page 23
Added a Note for the Generic Interrupt Controller	<a href="#">2.1.5 Supported standards and specifications</a> on page 24
Updated the Note for clarity	<a href="#">2.4.1 About power management</a> on page 32
<ul style="list-style-type: none"> <li>Updated the Note for VCPU and VSYS</li> <li>Updated information about Lock-mode and Split-mode</li> </ul>	<a href="#">2.4.2 Voltage domains</a> on page 32
<ul style="list-style-type: none"> <li>Changed shutdown to deactivated mode</li> <li>Added information about instructions for the thread</li> </ul>	<a href="#">2.4.4 Architectural clock gating modes</a> on page 35
Updated the section with more information about threads	<a href="#">2.4.6.5 Core dynamic retention mode</a> on page 41
Updated table for On and Core dynamic retention	<a href="#">2.4.6.7 Encoding for power modes</a> on page 42
Added deactivated mode	<a href="#">2.4.7 Thread power modes</a> on page 44
Updated information about power down for a thread	<a href="#">2.4.7.3 Deactivated mode</a> on page 44
<ul style="list-style-type: none"> <li>Updated table for SIMD dynamic retention</li> <li>Added a footnote in the table for Thread power modes</li> </ul>	<a href="#">2.4.8 Relationship between power modes and power domains</a> on page 45
Added information on DebugBlock	<a href="#">2.4.10 Debug over powerdown</a> on page 46
Clarification added to the TLB match description	<a href="#">2.5.3 TLB match process</a> on page 50

Change	Location
Added information about the memory access	<a href="#">2.5.4 Translation table walks</a> on page 51
Updated the encodings for all registers in the table	<a href="#">2.6.6 Direct access to internal memory</a> on page 65
Updated the L1 Data Cache Tag data format table	<a href="#">2.6.6.1 Encoding for tag and data in the L1 data cache</a> on page 66
<ul style="list-style-type: none"> <li>Added Thread ID table entry and updated the Size, AP/HYP, S2AP, Domain, DBM, and Parity descriptor fields for Tag RAM</li> <li>Updated the Parity descriptor field for Data RAM</li> </ul>	<a href="#">2.6.6.4 Main TLB RAM descriptor fields</a> on page 69
<ul style="list-style-type: none"> <li>Updated Domain description and added table entries for Thread ID and CnP in Tag RAM</li> <li>Updated tables entries for PA and Parity in Data RAM</li> </ul>	<a href="#">2.6.6.5 Walk cache descriptor fields</a> on page 70
<ul style="list-style-type: none"> <li>Added ThreadID and CnP entries and updated entries for IPA, Unused, and Parity in Tag RAM table</li> <li>Updated entries for PA, Unused, and Parity in Data RAM table</li> </ul>	<a href="#">2.6.6.6 IPA cache descriptor fields</a> on page 71
<ul style="list-style-type: none"> <li>Added information about error injection</li> <li>Changed the last error type in the table to Recoverable error and edited matching description</li> </ul>	<a href="#">2.8.7 Error injection</a> on page 80
Updated the section with information about the new Hybrid-mode and updated the figures	<a href="#">2.11.1 Implementing Split-Lock</a> on page 84
Error register names fixed to match the RAS specification	<a href="#">2.8.7 Error injection</a> on page 80  <a href="#">3.1.3 AArch64 IMPLEMENTATION DEFINED register summary</a> on page 96  <a href="#">3.1.4 AArch64 registers by functional group</a> on page 96  <a href="#">3.1.40 ERXPFPGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1</a> on page 141  <a href="#">3.1.41 ERXPFPGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1</a> on page 142  <a href="#">3.1.42 ERXPFPGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1</a> on page 144  <a href="#">3.2.1 Error System register summary</a> on page 191  <a href="#">3.2.5 ERROPFGCDN, Error Pseudo Fault Generation Count Down Register</a> on page 198  <a href="#">3.2.6 ERROPFGCTL, Error Pseudo Fault Generation Control Register</a> on page 198  <a href="#">3.2.7 ERROPFGF, Error Pseudo Fault Generation Feature Register</a> on page 200
Updated reset values for ID_AA64ISAR0_EL1 and MIDR_EL1	<a href="#">3.1.4 AArch64 registers by functional group</a> on page 96
Updated bit field description for ID_AA64ISAR0_EL1	<a href="#">3.1.53 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1</a> on page 152
Updated bit field description for ID_AA64MMFR1_EL1	<a href="#">3.1.56 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1</a> on page 157

Change	Location
Updated bit field description for ID_AA64MMFR2_EL1	<a href="#">3.1.57 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1</a> on page 159
Updated bit field description for LORID_EL1	<a href="#">3.1.61 LORID_EL1, LORegion ID Register, EL1</a> on page 164
Updated TCR registers	<a href="#">3.1.73 TCR_EL1, Translation Control Register, EL1</a> on page 178 <a href="#">3.1.74 TCR_EL2, Translation Control Register, EL2</a> on page 179 <a href="#">3.1.75 TCR_EL3, Translation Control Register, EL3</a> on page 180
Added information on the conditions for ERRORMISCO to be RAZ	<a href="#">3.2.4 ERRORMISCO, Error Record Miscellaneous Register 0</a> on page 195
Updated bit field description for ERR0PFGCTL	<a href="#">3.2.6 ERR0PFGCTL, Error Pseudo Fault Generation Control Register</a> on page 198
Added information on the conditions for ERR0STATUS to be RAZ	<a href="#">3.2.8 ERR0STATUS, Error Record Primary Status Register</a> on page 202
Added information on breakpoints and watchpoints	<a href="#">4.1.3.2 Breakpoints and watchpoints</a> on page 240
Updated the description for L2D_CACHE_REFILL and L3D_CACHE_REFILL	<a href="#">4.2.4 PMU events</a> on page 244
Updated the bit assignment figure for EDDFR	<a href="#">5.2.8 EDDFR, External Debug Feature Register</a> on page 282
Updated bit field description for EDPIDR0	<a href="#">5.2.11 EDPIDR0, External Debug Peripheral Identification Register 0</a> on page 286
Updated the bit assignment figure for PMCEID0_ELO	<a href="#">5.3.2 PMCEID0_ELO, Performance Monitors Common Event Identification Register 0, ELO</a> on page 293
Added clarification	<a href="#">5.5.2 PMPCSSR, PMU Snapshot Program Counter Sample Register</a> on page 312
Added clarification	<a href="#">5.5.3 PMCIDSSR, PMU Snapshot CONTEXTIDR_EL1 Sample Register</a> on page 313
Added clarification	<a href="#">5.5.4 PMCID2SSR, PMU Snapshot CONTEXTIDR_EL2 Sample Register</a> on page 314
Updated TRCDEVAFF0	<a href="#">5.6.21 TRCDEVAFF0, Device Affinity Register 0</a> on page 340
Updated TRCDEVAFF1	<a href="#">5.6.22 TRCDEVAFF1, Device Affinity Register 1</a> on page 340
Changed registers to Read-As-Zero/write ignore	<a href="#">5.6.47 TRCLAR, Software Lock Access Register</a> on page 364 <a href="#">5.6.48 TRCLSR, Software Lock Status Register</a> on page 364
Updated TRCVMIDCVR0	<a href="#">5.6.75 TRCVMIDCVR0, VMID Comparator Value Register 0</a> on page 388

**Table A-3: Differences between Issue 0100-00 and Issue 0100-01**

Change	Location
First non-confidential release for r1p0	-
Editorial changes	Entire manual
Added a progressive terminology commitment	<a href="#">Inclusive language commitment</a> on page 4
Changed rmpn to rpxy for consistency and clarity	<a href="#">1.1 Product revision status</a> on page 15
Added back information on conditions for the core to quiesce	<a href="#">2.4.4 Architectural clock gating modes</a> on page 35
Added mention of L1 instruction cache	<a href="#">2.4.6.6 Debug recovery mode</a> on page 42
Added a Comparator label column in table <a href="#">coredclsfault_p/r[7:0] bit assignments</a> on page 88	<a href="#">2.11.1.3 Core RAS reporting signals</a> on page 87



Change	Location
Added details on the GIC version to ID_AA64PFR0_EL1	<a href="#">3.1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1</a> on page 160
Added IESB bit field to the SCTLRL_EL2 bit assignment figure	<a href="#">3.1.71 SCTLRL_EL2, System Control Register, EL2</a> on page 175
Removed the UEO bit field from the ERROPFGE register	<a href="#">3.2.7 ERROPFGE, Error Pseudo Fault Generation Feature Register</a> on page 200
Removed references to T32 and A32 instructions	<a href="#">5.1.2 DBGBCRN_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265
Added PMCEID0 and PMCEID1 back in the register summary table	<a href="#">5.4.1 Memory-mapped PMU register summary</a> on page 301
Edited the reset value for TRCIDR1	<a href="#">5.6.1 ETM register summary</a> on page 317
Updated the description for TRCDEVAFF0	<a href="#">5.6.21 TRCDEVAFF0, Device Affinity Register 0</a> on page 340
Updated the description for TRCDEVAFF1	<a href="#">5.6.22 TRCDEVAFF1, Device Affinity Register 1</a> on page 340

**Table A-4: Differences between Issue 0100-01 and Issue 0101-02**

Change	Location
First non-confidential release for r1p1	-
Editorial changes	Entire manual
Changed power domain description	<a href="#">2.4.3 Power domains</a> on page 33
Added note to clear the interrupts of a core that is executing in lock-step as part of the powerdown sequence	<a href="#">2.4.6.7 Encoding for power modes</a> on page 42
Updated the PMU, PMUSS register list	<a href="#">5.6.1 ETM register summary</a> on page 317
Updated register description	<a href="#">5.1.2 DBGBCRN_EL1, Debug Breakpoint Control Registers, EL1</a> on page 265  <a href="#">5.1.3 DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1</a> on page 269
Updated Table 4-6 with 47, 4 CTI + 43 PMU	<a href="#">4.3.2 ETM trace unit generation options and resources</a> on page 259