



Arm[®] Neoverse[™] V2 Core

Revision: r0p1

Technical Reference Manual

Non-Confidential

Issue 02

Copyright © 2021–2022 Arm Limited (or its affiliates). 102375_0001_02_en
All rights reserved.



Arm® Neoverse™ V2 Core Technical Reference Manual

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.

Release Information

Document history

| Issue | Date | Confidentiality | Change |
|---------|-------------------|------------------|-------------------------------------|
| 0000-01 | 29 October 2021 | Confidential | First early access release for r0p0 |
| 0001-02 | 30 September 2022 | Non-Confidential | First early access release for r0p1 |

Proprietary Notice

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

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

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, 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 affiliates) 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 © 2021–2022 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|version 21.0)

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.

Contents

| | |
|--|-----------|
| 1. Introduction..... | 18 |
| 1.1 Product revision status..... | 18 |
| 1.2 Intended audience..... | 18 |
| 1.3 Conventions..... | 18 |
| 1.4 Useful resources..... | 20 |
| 2. The Neoverse™ V2 core..... | 22 |
| 2.1 Neoverse™ V2 core features..... | 22 |
| 2.2 Neoverse™ V2 core configuration options..... | 23 |
| 2.3 DSU-110 dependent features..... | 24 |
| 2.4 Supported standards and specifications..... | 25 |
| 2.5 Test features..... | 29 |
| 2.6 Design tasks..... | 29 |
| 2.7 Product revisions..... | 30 |
| 3. Technical overview..... | 31 |
| 3.1 Core components..... | 31 |
| 3.2 Interfaces..... | 35 |
| 3.3 Programmers model..... | 36 |
| 4. Clocks and resets..... | 37 |
| 5. Power management..... | 38 |
| 5.1 Voltage and power domains..... | 38 |
| 5.2 Architectural clock gating modes..... | 40 |
| 5.2.1 Wait for Interrupt and Wait for Event..... | 40 |
| 5.2.2 Low-power state behavior considerations..... | 41 |
| 5.3 Power control..... | 42 |
| 5.4 Core power modes..... | 42 |
| 5.4.1 On mode..... | 44 |
| 5.4.2 Off mode..... | 44 |
| 5.4.3 Emulated off mode..... | 45 |
| 5.4.4 Full retention mode..... | 45 |

| | |
|--|-----------|
| 5.4.5 Debug recovery mode..... | 45 |
| 5.4.6 Warm reset mode..... | 46 |
| 5.5 Performance and power management..... | 46 |
| 5.5.1 Maximum Power Mitigation Mechanism..... | 47 |
| 5.5.2 Performance Defined Power..... | 47 |
| 5.6 Neoverse™ V2 core powerup and powerdown sequence..... | 48 |
| 5.6.1 Managing RAS fault and error interrupts during the core powerdown..... | 48 |
| 5.7 Debug over powerdown..... | 49 |
| 6. Memory management..... | 51 |
| 6.1 Memory Management Unit components..... | 51 |
| 6.2 Translation Lookaside Buffer entry content..... | 53 |
| 6.3 Translation Lookaside Buffer match process..... | 53 |
| 6.4 Translation table walks..... | 54 |
| 6.5 Hardware management of the Access flag and dirty state..... | 55 |
| 6.6 Responses..... | 55 |
| 6.7 Memory behavior and supported memory types..... | 56 |
| 6.8 Page-based hardware attributes..... | 58 |
| 7. L1 instruction memory system..... | 59 |
| 7.1 L1 instruction cache behavior..... | 59 |
| 7.2 L1 instruction cache Speculative memory accesses..... | 60 |
| 7.3 Program flow prediction..... | 60 |
| 7.4 Instruction cache hardware coherency..... | 62 |
| 8. L1 data memory system..... | 63 |
| 8.1 L1 data cache behavior..... | 63 |
| 8.2 Instruction implementation in the L1 data memory system..... | 64 |
| 8.3 Internal exclusive monitor..... | 64 |
| 8.4 Data prefetching..... | 65 |
| 8.5 Write streaming mode..... | 66 |
| 9. L2 memory system..... | 68 |
| 9.1 L2 cache..... | 68 |
| 9.2 Support for memory types..... | 69 |
| 9.3 Transaction capabilities..... | 69 |
| 10. Direct access to internal memory..... | 70 |

| | |
|--|------------|
| 10.1 L1 cache encodings..... | 70 |
| 10.1.1 L1 instruction tag RAM returned data..... | 72 |
| 10.1.2 L1 instruction data RAM returned data..... | 73 |
| 10.1.3 L1 instruction TLB returned data..... | 73 |
| 10.1.4 L0 macro-operation RAM returned data..... | 75 |
| 10.1.5 L1 data tag RAM returned data..... | 76 |
| 10.1.6 L1 data data RAM returned data..... | 77 |
| 10.1.7 L1 data TLB returned data..... | 77 |
| 10.2 L2 cache encodings..... | 79 |
| 10.2.1 L2 tag RAM returned data..... | 81 |
| 10.2.2 L2 data RAM returned data..... | 83 |
| 10.2.3 L2 TLB RAM returned data..... | 84 |
| 10.2.4 L2 Victim RAM returned data..... | 86 |
| 11. RAS Extension support..... | 88 |
| 11.1 Cache protection behavior..... | 88 |
| 11.2 Error containment..... | 89 |
| 11.3 Fault detection and reporting..... | 90 |
| 11.4 Error detection and reporting..... | 90 |
| 11.4.1 Error reporting and performance monitoring..... | 90 |
| 11.5 Error injection..... | 91 |
| 11.6 AArch64 RAS registers..... | 92 |
| 12. GIC CPU interface..... | 93 |
| 12.1 Disable the GIC CPU interface..... | 93 |
| 12.2 AArch64 GIC registers..... | 94 |
| 13. Advanced SIMD and floating-point support..... | 95 |
| 14. Scalable Vector Extensions support..... | 96 |
| 15. System control..... | 97 |
| 15.1 AArch64 Generic system control registers..... | 97 |
| 16. Random number generator support..... | 100 |
| 16.1 AArch64 Random number control registers..... | 101 |
| 17. Debug..... | 102 |

| | |
|--|------------|
| 17.1 Supported debug methods..... | 103 |
| 17.2 Debug register interfaces..... | 104 |
| 17.2.1 Core interfaces..... | 105 |
| 17.2.2 Effects of resets on debug registers..... | 105 |
| 17.2.3 External access permissions to Debug registers..... | 105 |
| 17.2.4 Breakpoints and watchpoints..... | 106 |
| 17.3 Debug events..... | 106 |
| 17.4 Debug memory map and debug signals..... | 107 |
| 17.5 ROM table..... | 107 |
| 17.6 CoreSight component identification..... | 107 |
| 17.7 AArch64 Debug registers..... | 108 |
| 17.8 External Debug registers..... | 108 |
| 17.9 External CoreROM registers..... | 109 |
| 18. Performance Monitors Extension support..... | 110 |
| 18.1 Performance monitors events..... | 110 |
| 18.2 Performance monitors interrupts..... | 120 |
| 18.3 External register access permissions..... | 120 |
| 18.4 AArch64 Performance monitors registers..... | 121 |
| 18.5 External PMU registers..... | 121 |
| 19. Embedded Trace Extension support..... | 123 |
| 19.1 Trace unit resources..... | 124 |
| 19.2 Trace unit generation options..... | 124 |
| 19.3 Reset the trace unit..... | 125 |
| 19.4 Program and read the trace unit registers..... | 126 |
| 19.5 Trace unit register interfaces..... | 128 |
| 19.6 Interaction with the Performance Monitoring Unit and Debug..... | 128 |
| 19.7 ETE events..... | 129 |
| 19.8 AArch64 Trace registers..... | 129 |
| 19.9 External ETE registers..... | 130 |
| 20. Trace Buffer Extension support..... | 132 |
| 20.1 Program and read the trace buffer registers..... | 132 |
| 20.2 Trace buffer register interface..... | 132 |
| 21. Activity Monitors Extension support..... | 133 |

| | |
|--|------------|
| 21.1 Activity monitors access..... | 133 |
| 21.2 Activity monitors counters..... | 134 |
| 21.3 Activity monitors events..... | 134 |
| 21.4 AArch64 Activity monitors registers..... | 135 |
| 21.5 External AMU registers..... | 135 |
| 22. Statistical Profiling Extension support..... | 137 |
| 22.1 Statistical Profiling Extension events packet..... | 138 |
| 22.2 Statistical Profiling Extension data source packet..... | 139 |
| 22.3 AArch64 Statistical Profiling Extension registers..... | 139 |
| A. AArch64 registers..... | 140 |
| A.1 AArch64 Generic system control register summary..... | 140 |
| A.1.1 AIDR_EL1, Auxiliary ID Register..... | 142 |
| A.1.2 ACTLR_EL1, Auxiliary Control Register (EL1)..... | 143 |
| A.1.3 ACTLR_EL2, Auxiliary Control Register (EL2)..... | 145 |
| A.1.4 HACR_EL2, Hypervisor Auxiliary Control Register..... | 148 |
| A.1.5 ACTLR_EL3, Auxiliary Control Register (EL3)..... | 149 |
| A.1.6 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2)..... | 152 |
| A.1.7 LORID_EL1, LORegionID (EL1)..... | 154 |
| A.1.8 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register (EL1)..... | 156 |
| A.1.9 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register (EL3)..... | 158 |
| A.1.10 RMR_EL3, Reset Management Register (EL3)..... | 160 |
| A.1.11 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register (EL1)..... | 161 |
| A.1.12 IMP_CPUACTLR2_EL1, CPU Auxiliary Control Register 2 (EL1)..... | 163 |
| A.1.13 IMP_CPUACTLR3_EL1, CPU Auxiliary Control Register 3 (EL1)..... | 164 |
| A.1.14 IMP_CPUACTLR4_EL1, CPU Auxiliary Control Register 4 (EL1)..... | 166 |
| A.1.15 IMP_CPUECTLR_EL1, CPU Extended Control Register..... | 167 |
| A.1.16 IMP_CPUECTLR2_EL1, CPU Extended Control Register 2..... | 177 |
| A.1.17 IMP_CPUBUSQOS_EL1, CPU Bus QoS Register..... | 182 |
| A.1.18 IMP_CPUPPMCR3_EL3, CPU Power Performance Management Control Register..... | 184 |
| A.1.19 IMP_CPUPPMPDPCR_EL1, CPU Power Performance Management Control Register..... | 185 |
| A.1.20 IMP_CPUL2DIRTYLNCT_EL1, CPU L2 Dirty Line Count Register..... | 186 |
| A.1.21 IMP_CPUPWRCTLR_EL1, CPU Power Control Register..... | 187 |
| A.1.22 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register (EL1)..... | 190 |
| A.1.23 IMP_CPUACTLR5_EL1, CPU Auxiliary Control Register 5 (EL1)..... | 192 |
| A.1.24 IMP_CPUACTLR6_EL1, CPU Auxiliary Control Register 6 (EL1)..... | 194 |

| | |
|--|-----|
| A.1.25 IMP_CPUACTLR7_EL1, CPU Auxiliary Control Register 7 (EL1)..... | 195 |
| A.1.26 IMP_ATCR_EL2, CPU Auxiliary Translation Control Register (EL2)..... | 197 |
| A.1.27 IMP_AVTCR_EL2, CPU Virtualization Auxiliary Translation Control Register (EL2)..... | 199 |
| A.1.28 IMP_CPUPPMCR_EL3, CPU Power Performance Management Control Register..... | 201 |
| A.1.29 IMP_CPUMPMCR_EL3, CPU Power Performance Management Control Register..... | 203 |
| A.1.30 IMP_CPUPPMCR2_EL3, CPU Power Performance Management Control Register..... | 204 |
| A.1.31 IMP_CPUL2SDIRTYLNCT_EL3, CPU L2 Secure Dirty Line Count Register..... | 205 |
| A.1.32 IMP_CPUPDPTUNE_EL3, CPU Power Performance Management Control Register..... | 206 |
| A.1.33 IMP_CPUPPMCR4_EL3, CPU Power Performance Management Control Register..... | 208 |
| A.1.34 IMP_CPUPDPTUNE2_EL3, CPU Power Performance Management Control Register..... | 209 |
| A.1.35 IMP_CPUPPMCR5_EL3, CPU Power Performance Management Control Register..... | 211 |
| A.1.36 IMP_CPUMPMMTUNE_EL3, CPU Power Performance Management Control Register.... | 212 |
| A.1.37 IMP_CPUPPMCR6_EL3, CPU Power Performance Management Control Register..... | 213 |
| A.1.38 IMP_CPUACTLR_EL3, CPU Auxiliary Control Register (EL3)..... | 215 |
| A.1.39 IMP_ATCR_EL3, CPU Auxiliary Translation Control Register (EL2)..... | 216 |
| A.1.40 IMP_CPUPSELR_EL3, Selected Instruction Private Select Register..... | 218 |
| A.1.41 IMP_CPUPCR_EL3, Selected Instruction Private Control Register..... | 219 |
| A.1.42 IMP_CPUPOR_EL3, Selected Instruction Private Opcode Register..... | 221 |
| A.1.43 IMP_CPUPMR_EL3, Selected Instruction Private Mask Register..... | 222 |
| A.1.44 IMP_CPUPOR2_EL3, Selected Instruction Private Opcode Register 2..... | 223 |
| A.1.45 IMP_CPUPMR2_EL3, Selected Instruction Private Mask Register 2..... | 225 |
| A.1.46 IMP_CPUPFR_EL3, Selected Instruction Private Flag Register..... | 226 |
| A.1.47 FPCR, Floating-point Control Register..... | 227 |
| A.1.48 AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2)..... | 231 |
| A.1.49 AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2)..... | 233 |
| A.1.50 AFSR0_EL1, Auxiliary Fault Status Register 0 (EL1)..... | 236 |
| A.1.51 AFSR1_EL1, Auxiliary Fault Status Register 1 (EL1)..... | 238 |
| A.1.52 AFSR0_EL3, Auxiliary Fault Status Register 0 (EL3)..... | 241 |
| A.1.53 AFSR1_EL3, Auxiliary Fault Status Register 1 (EL3)..... | 242 |
| A.2 AArch64 Debug register summary..... | 244 |
| A.2.1 IMP_IDATA0_EL3, Instruction Register 0..... | 244 |
| A.2.2 IMP_IDATA1_EL3, Instruction Register 0..... | 245 |
| A.2.3 IMP_IDATA2_EL3, Instruction Register 0..... | 246 |
| A.2.4 IMP_DDATA0_EL3, Data Register 0..... | 247 |
| A.2.5 IMP_DDATA1_EL3, Data Register 1..... | 248 |
| A.2.6 IMP_DDATA2_EL3, Data Register 2..... | 249 |

| | |
|---|-----|
| A.3 AArch64 Random number control register summary..... | 250 |
| A.3.1 IMP_CPURNDBR_EL3, CPU Random Number Base Register..... | 250 |
| A.3.2 IMP_CPURNDPEID_EL3, CPU Random Number Packet Identification Register..... | 252 |
| A.4 AArch64 System instruction register summary..... | 253 |
| A.4.1 SYS_IMP_RAMINDEX, RAM Index..... | 253 |
| A.5 AArch64 Identification register summary..... | 255 |
| A.5.1 MIDR_EL1, Main ID Register..... | 255 |
| A.5.2 MPIDR_EL1, Multiprocessor Affinity Register..... | 257 |
| A.5.3 REVIDR_EL1, Revision ID Register..... | 259 |
| A.5.4 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0..... | 260 |
| A.5.5 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1..... | 263 |
| A.5.6 ID_AA64ZFR0_EL1, SVE Feature ID register 0..... | 264 |
| A.5.7 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0..... | 266 |
| A.5.8 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1..... | 268 |
| A.5.9 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0..... | 270 |
| A.5.10 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1..... | 271 |
| A.5.11 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0..... | 272 |
| A.5.12 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1..... | 275 |
| A.5.13 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0..... | 278 |
| A.5.14 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1..... | 280 |
| A.5.15 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2..... | 282 |
| A.5.16 CLIDR_EL1, Cache Level ID Register..... | 284 |
| A.5.17 GMID_EL1, Multiple tag transfer ID register..... | 288 |
| A.5.18 CTR_ELO, Cache Type Register..... | 289 |
| A.5.19 DCZID_ELO, Data Cache Zero ID register..... | 291 |
| A.5.20 MPAMIDR_EL1, MPAM ID Register (EL1)..... | 293 |
| A.5.21 IMP_CPUCFR_EL1, CPU Configuration Register..... | 294 |
| A.6 AArch64 Performance monitors register summary..... | 296 |
| A.6.1 PMMIR_EL1, Performance Monitors Machine Identification Register..... | 296 |
| A.6.2 PMCR_ELO, Performance Monitors Control Register..... | 298 |
| A.6.3 PMCEID0_ELO, Performance Monitors Common Event Identification register 0..... | 302 |
| A.6.4 PMCEID1_ELO, Performance Monitors Common Event Identification register 1..... | 309 |
| A.7 AArch64 GIC register summary..... | 316 |
| A.7.1 ICC_CTLR_EL1, Interrupt Controller Control Register (EL1)..... | 316 |
| A.7.2 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register..... | 320 |
| A.7.3 ICC_APOR0_EL1, Interrupt Controller Active Priorities Group 0 Registers..... | 323 |

| | |
|--|-----|
| A.7.4 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers..... | 324 |
| A.7.5 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Registers..... | 326 |
| A.7.6 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers..... | 327 |
| A.7.7 ICH_VTR_EL2, Interrupt Controller VGIC Type Register..... | 328 |
| A.7.8 ICC_CTLR_EL3, Interrupt Controller Control Register (EL3)..... | 330 |
| A.8 AArch64 Activity monitors register summary..... | 334 |
| A.8.1 AMEVTYPER10_ELO, Activity Monitors Event Type Registers 1..... | 335 |
| A.8.2 AMEVTYPER11_ELO, Activity Monitors Event Type Registers 1..... | 336 |
| A.8.3 AMEVTYPER12_ELO, Activity Monitors Event Type Registers 1..... | 337 |
| A.8.4 AMCFGR_ELO, Activity Monitors Configuration Register..... | 339 |
| A.8.5 AMCGCR_ELO, Activity Monitors Counter Group Configuration Register..... | 341 |
| A.8.6 AMEVTYPER00_ELO, Activity Monitors Event Type Registers 0..... | 342 |
| A.8.7 AMEVTYPER01_ELO, Activity Monitors Event Type Registers 0..... | 343 |
| A.8.8 AMEVTYPER02_ELO, Activity Monitors Event Type Registers 0..... | 344 |
| A.8.9 AMEVTYPER03_ELO, Activity Monitors Event Type Registers 0..... | 345 |
| A.9 AArch64 Trace register summary..... | 346 |
| A.9.1 TRCIDR8, ID Register 8..... | 347 |
| A.9.2 TRCIMSPEC0, IMP DEF Register 0..... | 348 |
| A.9.3 TRCIDR2, ID Register 2..... | 350 |
| A.9.4 TRCIDR3, ID Register 3..... | 352 |
| A.9.5 TRCIDR4, ID Register 4..... | 354 |
| A.9.6 TRCIDR5, ID Register 5..... | 356 |
| A.9.7 TRCIDR10, ID Register 10..... | 358 |
| A.9.8 TRCIDR11, ID Register 11..... | 360 |
| A.9.9 TRCIDR12, ID Register 12..... | 361 |
| A.9.10 TRCIDR13, ID Register 13..... | 362 |
| A.9.11 TRCIDR0, ID Register 0..... | 364 |
| A.9.12 TRCIDR1, ID Register 1..... | 366 |
| A.9.13 TRCCIDCVR0, Context Identifier Comparator Value Registers <n>..... | 368 |
| A.10 AArch64 MPAM register summary..... | 369 |
| A.10.1 MPAMVPMV_EL2, MPAM Virtual Partition Mapping Valid Register..... | 369 |
| A.10.2 MPAMVPM0_EL2, MPAM Virtual PARTID Mapping Register 0..... | 372 |
| A.10.3 MPAMVPM1_EL2, MPAM Virtual PARTID Mapping Register 1..... | 374 |
| A.10.4 MPAMVPM2_EL2, MPAM Virtual PARTID Mapping Register 2..... | 376 |
| A.10.5 MPAMVPM3_EL2, MPAM Virtual PARTID Mapping Register 3..... | 378 |
| A.10.6 MPAMVPM4_EL2, MPAM Virtual PARTID Mapping Register 4..... | 380 |

| | |
|--|------------|
| A.10.7 MPAMVPM5_EL2, MPAM Virtual PARTID Mapping Register 5..... | 382 |
| A.10.8 MPAMVPM6_EL2, MPAM Virtual PARTID Mapping Register 6..... | 384 |
| A.10.9 MPAMVPM7_EL2, MPAM Virtual PARTID Mapping Register 7..... | 386 |
| A.11 AArch64 RAS register summary..... | 388 |
| A.11.1 ERRIDR_EL1, Error Record ID Register..... | 388 |
| A.11.2 ERRSELR_EL1, Error Record Select Register..... | 390 |
| A.11.3 ERXFR_EL1, Selected Error Record Feature Register..... | 391 |
| A.11.4 ERXCTLR_EL1, Selected Error Record Control Register..... | 394 |
| A.11.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register..... | 397 |
| A.11.6 ERXADDR_EL1, Selected Error Record Address Register..... | 402 |
| A.11.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register..... | 404 |
| A.11.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register..... | 407 |
| A.11.9 ERXPFGCDN_EL1, Selected Pseudo-fault Generation Countdown register..... | 410 |
| A.11.10 ERXMISCO_EL1, Selected Error Record Miscellaneous Register 0..... | 412 |
| A.11.11 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1..... | 417 |
| A.11.12 ERXMISC2_EL1, Selected Error Record Miscellaneous Register 2..... | 419 |
| A.11.13 ERXMISC3_EL1, Selected Error Record Miscellaneous Register 3..... | 420 |
| A.12 AArch64 Statistical Profiling Extension register summary..... | 422 |
| A.12.1 PMBIDR_EL1, Profiling Buffer ID Register..... | 422 |
| A.12.2 PMSEVFR_EL1, Sampling Event Filter Register..... | 424 |
| A.12.3 PMSIDR_EL1, Sampling Profiling ID Register..... | 434 |
| B. External registers..... | 437 |
| B.1 External CoreROM register summary..... | 437 |
| B.1.1 COREROM_ROMENTRY0, Core ROM table Entry 0..... | 437 |
| B.1.2 COREROM_ROMENTRY1, Core ROM table Entry 1..... | 438 |
| B.1.3 COREROM_ROMENTRY2, Core ROM table Entry 2..... | 439 |
| B.1.4 COREROM_ROMENTRY3, Core ROM table Entry 3..... | 440 |
| B.1.5 COREROM_AUTHSTATUS, Core ROM table Authentication Status Register..... | 441 |
| B.1.6 COREROM_DEVARCH, Core ROM table Device Architecture Register..... | 442 |
| B.1.7 COREROM_DEVTYPE, Core ROM table Device Type Register..... | 443 |
| B.1.8 COREROM_PIDR4, Core ROM table Peripheral Identification Register 4..... | 444 |
| B.1.9 COREROM_PIDR0, Core ROM table Peripheral Identification Register 0..... | 445 |
| B.1.10 COREROM_PIDR1, Core ROM table Peripheral Identification Register 1..... | 446 |
| B.1.11 COREROM_PIDR2, Core ROM table Peripheral Identification Register 2..... | 447 |
| B.1.12 COREROM_PIDR3, Core ROM table Peripheral Identification Register 3..... | 448 |

| | |
|--|-----|
| B.1.13 COREROM_CIDR0, Core ROM table Component Identification Register 0..... | 448 |
| B.1.14 COREROM_CIDR1, Core ROM table Component Identification Register 1..... | 449 |
| B.1.15 COREROM_CIDR2, Core ROM table Component Identification Register 2..... | 450 |
| B.1.16 COREROM_CIDR3, Core ROM table Component Identification Register 3..... | 451 |
| B.2 External PPM register summary..... | 452 |
| B.2.1 CPUPPMCR, Power Performance Management Register..... | 452 |
| B.2.2 CPUPPMCR2, Power Performance Management Register..... | 453 |
| B.2.3 CPUPPMCR3, Power Performance Management Register..... | 453 |
| B.2.4 CPUPPMCR4, Power Performance Management Register..... | 454 |
| B.2.5 CPUPPMCR5, Power Performance Management Register..... | 455 |
| B.2.6 CPUPPMCR6, Power Performance Management Register..... | 456 |
| B.3 External PMU register summary..... | 456 |
| B.3.1 PMPCSSR, Snapshot Program Counter Sample Register..... | 457 |
| B.3.2 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register..... | 459 |
| B.3.3 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register..... | 459 |
| B.3.4 PMSSSR, PMU Snapshot Status Register..... | 460 |
| B.3.5 PMCCNTSR, PMU Cycle Counter Snapshot Register..... | 461 |
| B.3.6 PMEVCNTSR0, PMU Event Counter Snapshot Register..... | 462 |
| B.3.7 PMEVCNTSR1, PMU Event Counter Snapshot Register..... | 462 |
| B.3.8 PMEVCNTSR2, PMU Event Counter Snapshot Register..... | 463 |
| B.3.9 PMEVCNTSR3, PMU Event Counter Snapshot Register..... | 464 |
| B.3.10 PMEVCNTSR4, PMU Event Counter Snapshot Register..... | 465 |
| B.3.11 PMEVCNTSR5, PMU Event Counter Snapshot Register..... | 465 |
| B.3.12 PMSSCR, PMU Snapshot Capture Register..... | 466 |
| B.3.13 PMCFGR, Performance Monitors Configuration Register..... | 467 |
| B.3.14 PMCR_ELO, Performance Monitors Control Register..... | 468 |
| B.3.15 PMCEID0, Performance Monitors Common Event Identification register 0..... | 470 |
| B.3.16 PMCEID1, Performance Monitors Common Event Identification register 1..... | 473 |
| B.3.17 PMCEID2, Performance Monitors Common Event Identification register 2..... | 476 |
| B.3.18 PMCEID3, Performance Monitors Common Event Identification register 3..... | 480 |
| B.3.19 PMMIR, Performance Monitors Machine Identification Register..... | 483 |
| B.3.20 PMDEVARCH, Performance Monitors Device Architecture register..... | 484 |
| B.3.21 PMDEVID, Performance Monitors Device ID register..... | 485 |
| B.3.22 PMDEVTYPE, Performance Monitors Device Type register..... | 486 |
| B.3.23 PMPIDR4, Performance Monitors Peripheral Identification Register 4..... | 487 |
| B.3.24 PMPIDR0, Performance Monitors Peripheral Identification Register 0..... | 488 |

| | |
|--|-----|
| B.3.25 PMPIDR1, Performance Monitors Peripheral Identification Register 1..... | 489 |
| B.3.26 PMPIDR2, Performance Monitors Peripheral Identification Register 2..... | 490 |
| B.3.27 PMPIDR3, Performance Monitors Peripheral Identification Register 3..... | 491 |
| B.3.28 PMCIDR0, Performance Monitors Component Identification Register 0..... | 492 |
| B.3.29 PMCIDR1, Performance Monitors Component Identification Register 1..... | 493 |
| B.3.30 PMCIDR2, Performance Monitors Component Identification Register 2..... | 494 |
| B.3.31 PMCIDR3, Performance Monitors Component Identification Register 3..... | 495 |
| B.4 External Debug register summary..... | 496 |
| B.4.1 EDRCR, External Debug Reserve Control Register..... | 496 |
| B.4.2 EDACR, External Debug Auxiliary Control Register..... | 497 |
| B.4.3 EDPRCR, External Debug Power/Reset Control Register..... | 498 |
| B.4.4 MIDR_EL1, Main ID Register..... | 499 |
| B.4.5 EDPFR, External Debug Processor Feature Register..... | 500 |
| B.4.6 EDDFR, External Debug Feature Register..... | 502 |
| B.4.7 EDDEVARCH, External Debug Device Architecture register..... | 503 |
| B.4.8 EDDEVID2, External Debug Device ID register 2..... | 505 |
| B.4.9 EDDEVID1, External Debug Device ID register 1..... | 505 |
| B.4.10 EDDEVID, External Debug Device ID register 0..... | 506 |
| B.4.11 EDDEVTYPE, External Debug Device Type register..... | 507 |
| B.4.12 EDPIDR4, External Debug Peripheral Identification Register 4..... | 508 |
| B.4.13 EDPIDR0, External Debug Peripheral Identification Register 0..... | 509 |
| B.4.14 EDPIDR1, External Debug Peripheral Identification Register 1..... | 510 |
| B.4.15 EDPIDR2, External Debug Peripheral Identification Register 2..... | 511 |
| B.4.16 EDPIDR3, External Debug Peripheral Identification Register 3..... | 512 |
| B.4.17 EDCIDR0, External Debug Component Identification Register 0..... | 513 |
| B.4.18 EDCIDR1, External Debug Component Identification Register 1..... | 514 |
| B.4.19 EDCIDR2, External Debug Component Identification Register 2..... | 515 |
| B.4.20 EDCIDR3, External Debug Component Identification Register 3..... | 515 |
| B.5 External AMU register summary..... | 516 |
| B.5.1 AMEVTYPEP00, Activity Monitors Event Type Registers 0..... | 517 |
| B.5.2 AMEVTYPEP01, Activity Monitors Event Type Registers 0..... | 518 |
| B.5.3 AMEVTYPEP02, Activity Monitors Event Type Registers 0..... | 519 |
| B.5.4 AMEVTYPEP03, Activity Monitors Event Type Registers 0..... | 520 |
| B.5.5 AMEVTYPEP10, Activity Monitors Event Type Registers 1..... | 520 |
| B.5.6 AMEVTYPEP11, Activity Monitors Event Type Registers 1..... | 521 |
| B.5.7 AMEVTYPEP12, Activity Monitors Event Type Registers 1..... | 522 |

| | |
|--|-----|
| B.5.8 AMEVTYPER13, Activity Monitors Event Type Registers 1..... | 523 |
| B.5.9 AMCGCR, Activity Monitors Counter Group Configuration Register..... | 524 |
| B.5.10 AMCFGR, Activity Monitors Configuration Register..... | 525 |
| B.5.11 AMIIDR, Activity Monitors Implementation Identification Register..... | 527 |
| B.5.12 AMDEVARCH, Activity Monitors Device Architecture Register..... | 528 |
| B.5.13 AMDEVTYPE, Activity Monitors Device Type Register..... | 529 |
| B.5.14 AMPIDR4, Activity Monitors Peripheral Identification Register 4..... | 530 |
| B.5.15 AMPIDR0, Activity Monitors Peripheral Identification Register 0..... | 531 |
| B.5.16 AMPIDR1, Activity Monitors Peripheral Identification Register 1..... | 532 |
| B.5.17 AMPIDR2, Activity Monitors Peripheral Identification Register 2..... | 533 |
| B.5.18 AMPIDR3, Activity Monitors Peripheral Identification Register 3..... | 534 |
| B.5.19 AMCIDR0, Activity Monitors Component Identification Register 0..... | 535 |
| B.5.20 AMCIDR1, Activity Monitors Component Identification Register 1..... | 536 |
| B.5.21 AMCIDR2, Activity Monitors Component Identification Register 2..... | 537 |
| B.5.22 AMCIDR3, Activity Monitors Component Identification Register 3..... | 537 |
| B.6 External ETE register summary..... | 538 |
| B.6.1 TRCAUXCTLR, Auxillary Control Register..... | 539 |
| B.6.2 TRCIDR8, ID Register 8..... | 540 |
| B.6.3 TRCIDR9, ID Register 9..... | 541 |
| B.6.4 TRCIDR10, ID Register 10..... | 541 |
| B.6.5 TRCIDR11, ID Register 11..... | 542 |
| B.6.6 TRCIDR12, ID Register 12..... | 543 |
| B.6.7 TRCIDR13, ID Register 13..... | 544 |
| B.6.8 TRCIMSPECO, IMP DEF Register 0..... | 544 |
| B.6.9 TRCIDR0, ID Register 0..... | 545 |
| B.6.10 TRCIDR1, ID Register 1..... | 547 |
| B.6.11 TRCIDR2, ID Register 2..... | 548 |
| B.6.12 TRCIDR3, ID Register 3..... | 549 |
| B.6.13 TRCIDR4, ID Register 4..... | 551 |
| B.6.14 TRCIDR5, ID Register 5..... | 553 |
| B.6.15 TRCIDR6, ID Register 6..... | 554 |
| B.6.16 TRCIDR7, ID Register 7..... | 555 |
| B.6.17 TRCITCTRL, Integration Mode Control Register..... | 555 |
| B.6.18 TRCCLAIMSET, Claim Tag Set Register..... | 556 |
| B.6.19 TRCCLAIMCLR, Claim Tag Clear Register..... | 557 |
| B.6.20 TRCDEVARCH, Device Architecture Register..... | 558 |

| | |
|--|------------|
| B.6.21 TRCDEVID2, Device Configuration Register 2..... | 559 |
| B.6.22 TRCDEVID1, Device Configuration Register 1..... | 560 |
| B.6.23 TRCDEVID, Device Configuration Register..... | 561 |
| B.6.24 TRCDEVTYPE, Device Type Register..... | 562 |
| B.6.25 TRCPIDR4, Peripheral Identification Register 4..... | 563 |
| B.6.26 TRCPIDR5, Peripheral Identification Register 5..... | 563 |
| B.6.27 TRCPIDR6, Peripheral Identification Register 6..... | 564 |
| B.6.28 TRCPIDR7, Peripheral Identification Register 7..... | 565 |
| B.6.29 TRCPIDR0, Peripheral Identification Register 0..... | 566 |
| B.6.30 TRCPIDR1, Peripheral Identification Register 1..... | 566 |
| B.6.31 TRCPIDR2, Peripheral Identification Register 2..... | 567 |
| B.6.32 TRCPIDR3, Peripheral Identification Register 3..... | 568 |
| B.6.33 TRCCIDR0, Component Identification Register 0..... | 569 |
| B.6.34 TRCCIDR1, Component Identification Register 1..... | 570 |
| B.6.35 TRCCIDR2, Component Identification Register 2..... | 571 |
| B.6.36 TRCCIDR3, Component Identification Register 3..... | 571 |
| C. Document revisions..... | 573 |
| C.1 Revisions..... | 573 |

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.

| Convention | Use |
|----------------------------|---|
| <i>italic</i> | Citations. |
| bold | 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. |
| 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. |
| <and> | Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <pre>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></pre> |

| Convention | Use |
|-----------------------|--|
| SMALL CAPITALS | Terms that have specific technical meanings as defined in the <i>Arm® Glossary</i> . For example, IMPLEMENTATION DEFINED , IMPLEMENTATION SPECIFIC , UNKNOWN , and UNPREDICTABLE . |



Recommendations. Not following these recommendations might lead to system failure or damage.



Requirements for the system. Not following these requirements might result in system failure or damage.



Requirements for the system. Not following these requirements will result in system failure or damage.



An important piece of information that needs your attention.



A useful tip that might make it easier, better or faster to perform a task.

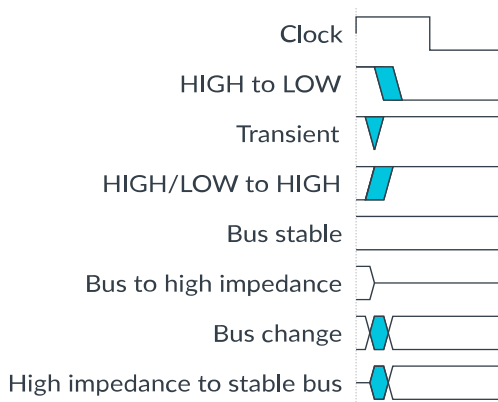


A reminder of something important that relates to the information you are reading.

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 Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Table 1-2: Arm publications

| Document Name | Document ID | Licensee only |
|--|-------------|---------------|
| Arm® Neoverse™ V2 Core Configuration and Integration Manual | 102393 | Yes |
| Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual | 101381 | No |
| Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual | 101382 | Yes |

| Document Name | Document ID | Licensee only |
|--|-------------|---------------|
| <i>Arm® Architecture Reference Manual for A-profile architecture</i> | DDI 0487 | No |
| <i>Arm® Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM) for Armv8-A</i> | DDI 0598 | No |
| <i>Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile</i> | DDI 0608 | No |
| <i>Arm® Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile</i> | DDI 0587 | No |
| <i>Arm® Architecture Reference Manual Supplement The Scalable Vector Extension (SVE) for Armv8-A</i> | DDI 0584 | No |
| <i>AMBA® 5 CHI Architecture Specification</i> | IHI 0050 | No |
| <i>Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4</i> | IHI 0069 | No |
| <i>Arm® CoreSight™ Architecture Specification v3.0</i> | IHI 0029 | No |
| <i>Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual</i> | 101088 | No |

Table 1-3: Other publications

| Document ID | Document Name |
|-------------|---------------|
| - | - |

**Note**

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>

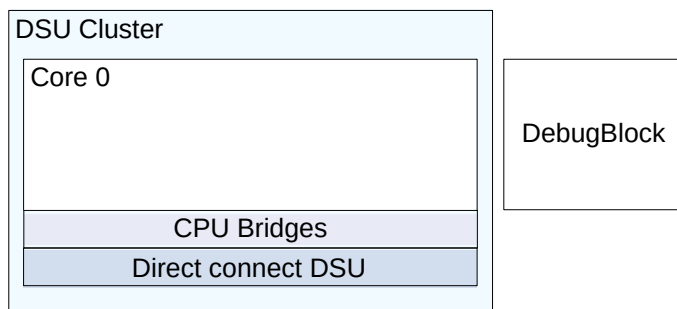
2. The Neoverse™ V2 core

The Neoverse™ V2 core is a high-performance and low-power product that implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Armv8-A architectures up to Arm®v8.5-A.

The Neoverse™ V2 core is implemented inside a DSU cluster and is always connected to the DSU-110. The Neoverse™ V2 core supports Direct connect only. For more information on the DSU Direct connect, see the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*.

The following figure shows an example configuration with one Neoverse™ V2 core that is implemented as a single core in a DSU cluster which is configured for Direct connect, without the L3 cache, snoop filter, or *Snoop Control Unit* (SCU) logic present.

Figure 2-1: Neoverse™ V2 example configuration



This manual applies to the Neoverse™ V2 core only. Read this manual together with the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for detailed information about the DSU-110.

This manual does not provide a complete list of registers. Read this manual together with the [Arm® Architecture Reference Manual for A-profile architecture](#).

2.1 Neoverse™ V2 core features

The Neoverse™ V2 core can be used in a standalone DSU configuration, which is configured in Direct connect mode.

Core features

- Implementation of the Armv9-A A64 instruction set.
- AArch64 Execution state at all Exception levels, EL0 to EL3.
- *Memory Management Unit* (MMU)

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

Cache features

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

Debug features

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

Related information

[3. Technical overview](#) on page 31

2.2 Neoverse™ V2 core configuration options

You can choose the options that fit your implementation needs at build-time configuration.

The Neoverse™ V2 core configuration options include:

Cryptographic Extension

You can configure your implementation with or without the Cryptographic Extension.

Coherent instruction cache

You can configure your implementation with or without support for coherent instruction cache.

Random Number Generator

You can configure your implementation with or without support for Armv8.5-RNG.

L2 cache size

You can configure the L2 cache to be 1MB or 2MB.

CoreSight™ Embedded Logic Analyzer (ELA)

You can include support for integrating ELA-600 as a separate licensable product.

Size of the ATB FIFO depth in the core ELA

You can configure the size of the AMBA® Trace Bus (ATB) FIFO to be 4, 8, 16, 32, or 64.

Timing closure

You can configure the L2 data cache RAMs timing behavior. For more information, see the *demeter.yaml* file section in the *RTL configuration process* chapter of the *Arm® Neoverse™ V2 Core Configuration and Integration Manual*.

For detailed configuration options and guidelines, see the *RTL configuration process* chapter of the *Arm® Neoverse™ V2 Core Configuration and Integration Manual*.

2.3 DSU-110 dependent features

Support for some DSU-110 features and behaviors depends on whether your licensed core supports a particular feature.

The following table describes which DSU-110 dependent features are supported in your Neoverse™ V2 core.

Table 2-1: Neoverse™ V2 core features that have a dependency on the DSU-110

| Feature | Supported in the Neoverse™ V2 core | Dependency on the DSU-110 |
|----------------------------|------------------------------------|--|
| Direct connect | Only supports Direct connect | Direct connect support at the cluster level only applies when your licensed core also supports Direct connect. Direct connect is intended for large systems where there are many cores. |
| Core included in a complex | No | Affects the cluster configuration and external signals. |

| Feature | Supported in the Neoverse™ V2 core | Dependency on the DSU-110 |
|--|------------------------------------|---|
| Cryptographic Extension | Yes | Affects the external signals of the DSU-110. |
| SMCRYPTODISABLE signal supported | Yes | For more information, see the <i>Cryptographic extension support in the Neoverse™ V2 core</i> chapter of the <i>Arm® Neoverse™ V2 Core Cryptographic Extension Technical Reference Manual</i> . See the <i>DynamlQ™ Shared Unit-110 signals</i> section in the <i>Functional integration</i> chapter of the <i>Arm® DynamlQ™ Shared Unit-110 Configuration and Integration Manual</i> for the connection information of these signals. |
| Maximum Power Mitigation Mechanism (MPMM) | Yes | Affects the external signals of the DSU-110. See the <i>DynamlQ™ Shared Unit-110 signals</i> section in the <i>Functional integration</i> chapter of the <i>Arm® DynamlQ™ Shared Unit-110 Configuration and Integration Manual</i> for the connection information of these signals. |
| Performance Defined Power (PDP) feature | Yes | |
| DISPBLKy signal supported | Yes | |
| Statistical Profiling Extension (SPE) architecture | Yes | |
| Physical Address (PA) width | 48-bit | Affects the CHI and AXI master port bus widths. For more details, see the following chapters of the <i>Arm® DynamlQ™ Shared Unit-110 Technical Reference Manual</i> : <ul style="list-style-type: none"> • <i>CHI master interface</i> • <i>AXI master interface</i> |

2.4 Supported standards and specifications

The Neoverse™ V2 core implements the Arm®v9.0-A architecture and supports all previous Armv8-A architectures up to Arm®v8.5-A. It also implements specific Arm architecture extensions and supports interconnect, interrupt, timer, debug, and trace architectures.

The Neoverse™ V2 core supports AArch64 at all Exception levels, EL0 to EL3, and supports all mandatory features of each architecture version.

The following tables show, for each Armv8-A architecture version, the optional features that the Neoverse™ V2 core supports.

Table 2-2: Armv8.0-A optional feature support in the Neoverse™ V2 core

| Feature | Status | Notes |
|-------------------------|---------------------------------------|--|
| Cryptographic Extension | Supported using a configurable option | For more information, see the <i>Arm® Neoverse™ V2 Core Cryptographic Extension Technical Reference Manual</i> . This extension is licensed separately and access to the documentation is restricted by contract with Arm. |

| Feature | Status | Notes |
|--|-----------|--|
| FEAT_AdvSIMD, Advanced Single Instruction Multiple Data (SIMD) Extension | Supported | For more information, see 13. Advanced SIMD and floating-point support on page 95. |
| FEAT_FP, Floating-point Extension | Supported | |
| FEAT_PMUv3, Performance Monitors Extension | Supported | See the Arm® Architecture Reference Manual for A-profile architecture for information on this feature. |
| FEAT_DGH, Data Gathering Hint | Supported | Adds the Data Gathering Hint instruction to the hint space. |

Table 2-3: Arm®v8.1-A optional feature support in the Neoverse™ V2 core

| Feature | Status | Notes |
|---|-----------|--|
| FEAT_HAFDBS, Hardware Management of the Access Flag and Dirty State | Supported | See the Arm® Architecture Reference Manual for A-profile architecture for information on these features. |
| FEAT_VMID16, 16-bit Virtual Machine Identifier (VMID) | Supported | |
| FEAT_PAN3, Support for SCTLRL_ELx.EPAN | Supported | |

Table 2-4: Arm®v8.2-A optional feature support in the Neoverse™ V2 core

| Feature | Status | Notes |
|--|--|--|
| FEAT_HPDS2, Translation Table Page-Based Hardware Attributes | Supported | See the Arm® Architecture Reference Manual for A-profile architecture for information on these features. |
| FEAT_PCSRv8p2, PC Sample-based Profiling | Supported | |
| FEAT_SHA512, Advanced SIMD SHA512 instructions | Supported as part of Armv8-A Cryptographic Extension | See the Arm® Architecture Reference Manual for A-profile architecture for information on these features. |
| FEAT_SHA3, Advanced SIMD SHA3 instructions | | |
| FEAT_SM3, Advanced SIMD SM3 instructions | | |
| FEAT_SM4, Advanced SIMD SM4 instructions | | |
| FEAT_BF16, 16-bit floating-point instructions | Supported | |
| FEAT_I8MM, Int8 Matrix Multiply instructions | Supported | |
| FEAT_MPAM, Memory System Resource Partitioning and Monitoring (MPAM) Extension | Supported | See the Arm® Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM) , for Armv8-A for information on this extension. |
| FEAT_SVE, Scalable Vector Extension | Supported | See 14. Scalable Vector Extensions support on page 96 and the Arm® Architecture Reference Manual for A-profile architecture for information on this extension. |
| FEAT_LPA, Large Physical Address (PA) and Intermediate PA (IPA) Support | Not supported | - |

| Feature | Status | Notes |
|--|---------------|---|
| FEAT_LVA, Large Virtual Address(VA) Support | Not supported | - |
| FEAT_LSMAOC, Load/Store Multiple Atomicity and Ordering Controls | Not supported | - |
| FEAT_AA32HPD, AArch32 Hierarchical Permission Disables | Not supported | - |
| FEAT_SPE, Statistical Profiling Extension | Supported | For more information on this extension, see 22. Statistical Profiling Extension support on page 137 and the Arm® Architecture Reference Manual for A-profile architecture . |

Table 2-5: Arm®v8.3-A optional feature support in the Neoverse™ V2 core

| Feature | Status | Notes |
|--|-----------|--|
| FEAT_NV, Nested Virtualization | Supported | - |
| FEAT_CCIDX, Extended Cache Index | Supported | See the Arm® Architecture Reference Manual for A-profile architecture for information on this feature. |
| FEAT_PAuth2, Pointer Authentication Enhancements | Supported | - |
| FEAT_FPAC, Faulting on AUT* instructions | Supported | - |

Table 2-6: Arm®v8.4-A optional feature support in the Neoverse™ V2 core

| Feature | Status | Notes |
|--|-----------|--|
| FEAT_AMUv1, Activity Monitors Extension | Supported | See the Arm® Architecture Reference Manual for A-profile architecture for information on this feature. |
| FEAT_NV2, Enhanced support for Nested Virtualization | Supported | - |

Table 2-7: Arm®v8.5-A optional feature support in the Neoverse™ V2 core

| Feature | Status | Notes |
|--|---------------------------------------|--|
| FEAT_MTE and FEAT_MTE2, Memory Tagging Extension | Supported | The Neoverse™ V2 core always implements MTE and therefore is compliant with the CHI Issue E protocol. See the <i>CHI master interface</i> chapter in the <i>Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual</i> for information on CHI.E commands inferred by MTE. |
| FEAT_MTE3, MTE Asymmetric Fault Handling | Supported | MTE enhancement. |
| FEAT_RNG, Random Number Generator | Supported using a configurable option | - |
| FEAT_ExS, Context Synchronization and Exception Handling | Not supported | - |

The following table shows the Arm®v9.0-A features that the Neoverse™ V2 core supports.

Table 2-8: Arm®v9.0-A feature support in the Neoverse™ V2 core

| Feature | Status | Notes |
|--|---------------|---|
| FEAT_SVE2, Scalable Vector Extension 2 | Supported | See 14. Scalable Vector Extensions support on page 96. |
| FEAT_SVE_AES, Scalable Vector AES Instructions | Supported | See the <i>Arm® Architecture Reference Manual Supplement, The Scalable Vector Extension</i> for more information. |
| FEAT_SVE_BitPerm, Scalable Vector Bit Permutes | Supported | |
| FEAT_SVE_PMULL, Scalable Vector Polynomial Multiply Instructions which Generate a 128-bit Result | Supported | |
| FEAT_SVE_SHA3, Scalable Vector SHA3 Instructions | Supported | |
| FEAT_SVE_SM4, Scalable Vector SM Instructions | Supported | |
| FEAT_ETE, <i>Embedded Trace Extension</i> (ETE) | Supported | See 19. Embedded Trace Extension support on page 123. |
| FEAT_TRBE, <i>TRace Buffer Extension</i> (TRBE) | Supported | See 20. Trace Buffer Extension support on page 132. |
| FEAT_TME, <i>Transactional Memory Extension</i> (TME) | Not supported | - |

The following table shows the other standards and specifications that the Neoverse™ V2 core supports.

Table 2-9: Other standards and specifications support in the Neoverse™ V2 core

| Standard or specification | Version | Notes |
|--|---------|--|
| FEAT_GICv4p1, <i>Generic Interrupt Controller</i> (GIC) 4.1 | GICv4.1 | See the <i>Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4</i> for more information. |
| Debug | - | Arm®v9.0-A architecture implemented with Arm®v8.4-A Debug architecture support and Arm®v8.3-A debug over powerdown support See the <i>Arm®v8.5 Debug Architecture</i> for information on this architecture. |
| FEAT_RASv1p1, <i>Reliability, Availability, and Serviceability</i> (RAS) Extensions | v1.1 | All extensions up to Arm®v9.0-A with <i>Error Correcting Code</i> (ECC) configured. See 11. RAS Extension support on page 88 for more information on the implementation of this extension in the core. |
| CoreSight | v3.0 | See the <i>Arm® CoreSight™ Architecture Specification v3.0</i> for more information. |
| FEAT_ECBHB, <i>Exploitative Control using Branch History Buffer</i> information between exception levels | - | The branch history information created in a context before an exception to a higher exception level, using AArch64, cannot be used by code before that exception. This prevents exploitative control of the execution of any indirect branches in code in a different context after the exception. |

Related information

[3.1 Core components](#) on page 31

2.5 Test features

The Neoverse™ V2 core provides test signals that enable the use of both *Automatic Test Pattern Generation* (ATPG) and *Memory Built-In Self Test* (MBIST) to test the core logic and memory arrays.

The Neoverse™ V2 core includes an *Automatic Test Pattern Generation* (ATPG) test interface that provides signals to control the *Design For Test* (DFT) features of the core. To prevent problems with DFT implementation, carefully consider usage of these signals.

Arm also provides *Memory Built-In Self Test* (MBIST) interfaces that enable you to test the RAMs at operational frequency. You can add your own MBIST controllers to automatically generate test patterns and perform result comparisons. Optionally, you can use your EDA MBIST interfaces instead of the supplied Arm interfaces.

See the *Design for Test integration guidelines* chapter of the *Arm® Neoverse™ V2 Core Configuration and Integration Manual* for the list of test signals and information on their usage. Also see the *Design for Test integration guidelines* chapter of the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for the list of external scan control signals.

2.6 Design tasks

The Neoverse™ V2 core is delivered as a synthesizable RTL description in SystemVerilog HDL. Before you can use the Neoverse™ V2 core, you must implement it, integrate it, and program it. Implementation and integration choices affect the behavior and features of the core.

A different party can perform each of the following tasks:

Implementation

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

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

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

Integration

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

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

Software programming

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

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

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

See the *RTL configuration process* chapter of the *Arm® Neoverse™ V2 Core Configuration and Integration Manual* and *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for implementation options. See also the *Functional integration* chapter of the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for signal descriptions.

2.7 Product revisions

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

Table 2-10: Product revisions

| Revision | Notes |
|----------|----------------------------|
| r0p0 | First early access release |
| r0p1 | First early access release |

Changes in functionality that have an impact on the documentation also appear in [C.1 Revisions](#) on page 573.

3. Technical overview

All components in the Neoverse™ V2 core are always present. These components are designed to make the Neoverse™ V2 core a high-performance core.

The main blocks include:

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

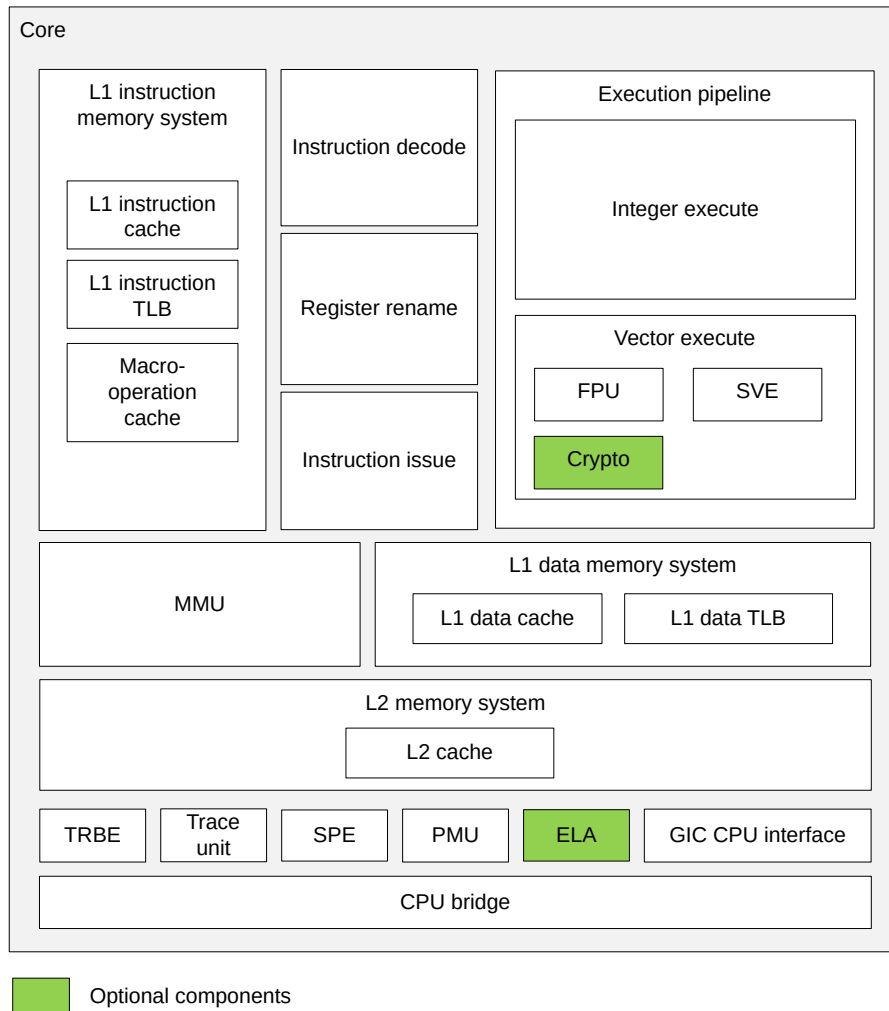
The Neoverse™ V2 core interfaces with the DSU-110 through the CPU bridge.

The Neoverse™ V2 core implements the Arm®v9.0-A architecture and supports all previous Armv8-A architectures up to Arm®v8.5-A. The programmers model and the architecture features implemented, such as the Generic Timer, are compliant with the standards in [2.4 Supported standards and specifications](#) on page 25.

3.1 Core components

The Neoverse™ V2 core includes components designed to make it a high-performance and low-power product. The Neoverse™ V2 core includes a CPU bridge that connects the core to the DSU-110. The DSU-110 connects the core to an external memory system and the rest of the SoC.

The following figure shows the Neoverse™ V2 core components.

Figure 3-1: Neoverse™ V2 core components

L1 instruction memory system

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

The L1 instruction memory system includes:

- A 64KB, 4-way set associative L1 instruction cache with 64-byte cache lines.
- A fully associative L1 instruction *Translation Lookaside Buffer* (TLB) with native support for 4KB, 16KB, 64KB, and 2MB page sizes.
- A 1536-entry, 4-way skewed associative *L0 Macro-OP* (MOP) cache, which contains decoded and optimized instructions for higher performance.
- A dynamic branch predictor.

Instruction decode

The instruction decode unit decodes AArch64 instructions into internal format.

Register rename

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

Instruction issue

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

Integer execute

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

Vector execute

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

Advanced SIMD and floating-point support

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

Cryptographic Extension

The Cryptographic Extension is optional in the Neoverse™ V2 cores. The Cryptographic Extension adds new instructions to the Advanced SIMD and the *Scalable Vector Extension* (SVE) instruction sets that accelerate:

- *Advanced Encryption Standard* (AES) encryption and decryption.
- The *Secure Hash Algorithm* (SHA) functions SHA-1, SHA-2, SHA-3, SHA-224, SHA-256, SHA-384, and SHA-512.
 - The SVE2 versions of the SHA-3 instructions EOR3, XAR, and BCAX are supported even when CRYPTO support is not configured.
- Armv8.2-SM SM3 hash function and SM4 encryption and decryption instructions.
- Finite field arithmetic that is used in algorithms such as Galois/Counter Mode and Elliptic Curve Cryptography.



Note

The optional Cryptographic Extension is not included in the base product. Arm supplies the Cryptographic Extension under an additional license to the Neoverse™ V2 core license.

Scalable Vector Extension

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

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



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

L1 data memory system

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

The L1 data memory system includes:

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

Memory Management Unit

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

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

L2 memory system

The L2 memory system includes the L2 cache. The L2 cache is private to the core and is 8-way set associative. You can configure its RAM size to be 1MB or 2MB. The L2 memory system is connected to the DSU-110 through an asynchronous CPU bridge.

Embedded Trace Extension and Trace Buffer Extension

The Neoverse™ V2 core supports a range of debug, test, and trace options including a trace unit and a trace buffer.

The Neoverse™ V2 core also includes a ROM table that contains a list of components in the system. Debuggers can use the ROM table to determine which CoreSight components are implemented.

All the debug and trace components of the Neoverse™ V2 core are described in this manual. For more information about the *Embedded Logic Analyzer* (ELA), see the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual*.

Statistical Profiling Extension

The Neoverse™ V2 core implements the *Statistical Profiling Extension* (SPE) to the Arm®v8.4-A architecture. The SPE provides a statistical view of the performance characteristics of executed instructions that software writers can use to optimize their code for better performance.

Performance Monitoring Unit

The *Performance Monitoring Unit* (PMU) provides 6 performance monitors that can be configured to gather statistics on the operation of each core and the memory system. The information can be used for debug and code profiling.

Activity Monitoring Unit

The Neoverse™ V2 core implements the Activity Monitors Extension to the Arm®v8.4-A architecture. Activity monitors in the *Activity Monitoring Unit* (AMU) provide useful information for system power management and persistent monitoring.

GIC CPU interface

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

CPU bridge

In a cluster, there is one CPU bridge between each Neoverse™ V2 core and the DSU-110.

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

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

Related information

- [6. Memory management](#) on page 51
- [7. L1 instruction memory system](#) on page 59
- [8. L1 data memory system](#) on page 63
- [9. L2 memory system](#) on page 68
- [12. GIC CPU interface](#) on page 93
- [13. Advanced SIMD and floating-point support](#) on page 95
- [18. Performance Monitors Extension support](#) on page 110
- [19. Embedded Trace Extension support](#) on page 123

3.2 Interfaces

The DSU-110 manages all Neoverse™ V2 core external interfaces to the *System on Chip* (SoC).

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

3.3 Programmers model

The Neoverse™ V2 core implements the Arm®v9.0-A architecture and supports all Armv8-A architectures up to Arm®v8.5-A. The Neoverse™ V2 core supports the AArch64 Execution state at all Exception levels, EL0 to EL3.

See the [Arm® Architecture Reference Manual for A-profile architecture](#) for more information about the programmers model.

Related information

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

4. Clocks and resets

The Neoverse™ V2 core supports hierarchical clock gating to provide dynamic power savings. The core also supports Warm and Cold resets.

Each Neoverse™ V2 core has a single clock domain and receives a single clock input. This clock input is gated by an architectural clock gate in the CPU bridge.

In addition, the Neoverse™ V2 core implements extensive clock gating that includes:

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

The Neoverse™ V2 core receives the following reset signals from the DSU-110 side of the CPU bridge:

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

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

5. Power management

The Neoverse™ V2 core provides mechanisms to control both dynamic and static power dissipation.

The dynamic power management includes the following features:

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

The static power management includes the following features:

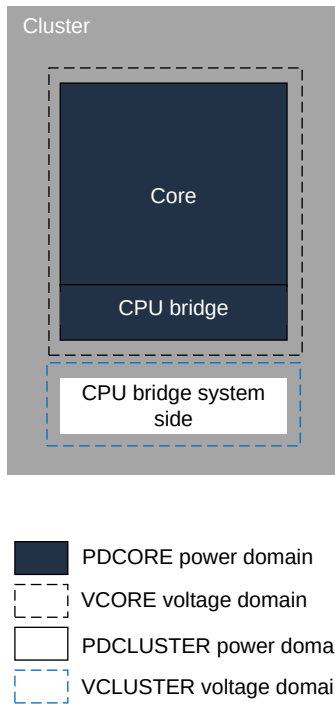
- Powerdown
- Dynamic retention, a low-power mode that retains the register and RAM state

5.1 Voltage and power domains

The DSU-110 *Power Policy Units* (PPUs) control power management for the Neoverse™ V2 core. The core supports one power domain, PDCORE, and one system power domain, PDCLUSTER. Similarly, it supports one core voltage domain, VCORE, and one cluster system voltage domain, VCLUSTER. The power and voltage domains have the same boundaries.

The PDCORE power domain contains all Neoverse™ V2 core logic and part of the core asynchronous bridge that belongs to the VCORE domain. The PDCLUSTER power domain contains the part of the CPU bridge that belongs to the VCLUSTER domain.

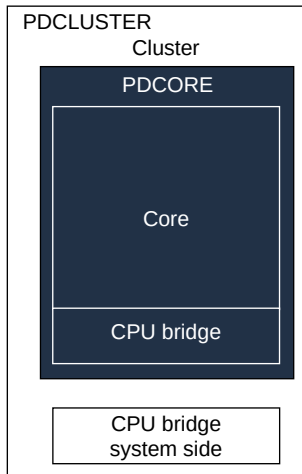
The following figure shows the Neoverse™ V2 core power domain and voltage domain. It also shows the cluster power domain and voltage domain that cover the system side of the CPU bridge.

Figure 5-1: Neoverse™ V2 core voltage and power domains

You can tie the VCORE and VCLUSTER voltage domains to the same supply if one of the following is true:

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

The following figure shows an example of the power domains with a Neoverse™ V2 core in a cluster.

Figure 5-2: Core power domains in a cluster with a Neoverse™ V2 core

Clamping cells between power domains are inferred through power intent files rather than instantiated in the RTL. See the *Power management* chapter of the *Arm® Neoverse™ V2 Core Configuration and Integration Manual* for more information.

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

5.2 Architectural clock gating modes

The *Wait For Interrupt* (WFI) and *Wait For Event* (WFE) instructions put the core into a low-power mode. These instructions work by architecturally disabling the clock at the top of the clock tree. The core remains fully powered and retains all state.

5.2.1 Wait for Interrupt and Wait for Event

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

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

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



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

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

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

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

See the [Arm® Architecture Reference Manual for A-profile architecture](#) for more information about entering low-power state and wakeup events.

5.2.2 Low-power state behavior considerations

You must consider how certain events affect the *Wait for Interrupt* (WFI) and *Wait for Event* (WFE) low-power state behavior of the Neoverse™ V2 core.

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

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



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

When the core enters WFI or WFE state, the core clock is gated.

See the [Arm® Architecture Reference Manual for A-profile architecture](#) for more information about WFI and WFE.

5.3 Power control

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

The core has its own PPU to control its own core power domain. In addition, there is a PPU for the cluster.

The PPUs decide and request any change in power mode. The Neoverse™ V2 core then performs any actions necessary to reach the requested power mode. For example, the core might gate clocks, clean caches, or disable coherency before it accepts the request.

For more information about the PPUs for the cluster and the cores, see the following chapters in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*:

- *Power management*
- *Power and reset control with Power Policy Units*

5.4 Core power modes

The Neoverse™ V2 core power domain has a defined set of power modes and corresponding legal transitions between these modes.

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

The following table shows the supported Neoverse™ V2 power modes.



Power modes that are not shown in the following table are not supported and must not occur.

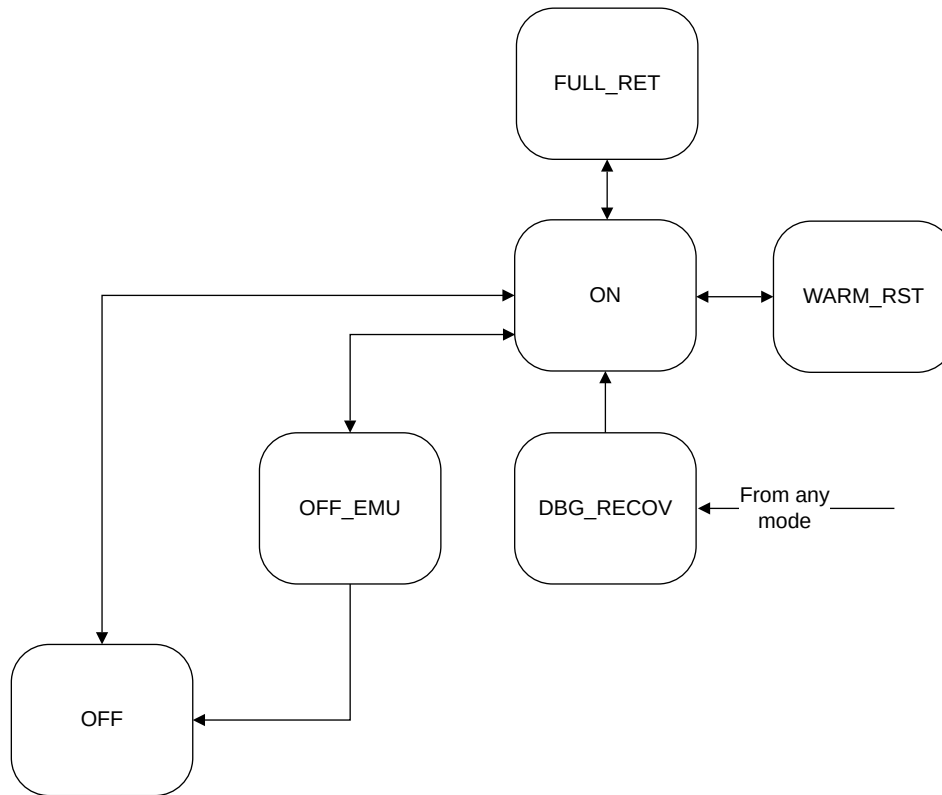
Table 5-1: Neoverse™ V2 core power modes

| Power mode | Short name | Power state |
|----------------|------------|--|
| On | ON | The core is powered up and active. |
| Full retention | FULL_RET | <p>The core is in retention. In this mode, only power that is required to retain register and RAM state is available. The core is not operational.</p> <p>A core must be in <i>Wait for Interrupt</i> (WFI) or <i>Wait for Event</i> (WFE) low-power state before it enters this mode.</p> |
| Off | OFF | The core is powered down. |

| Power mode | Short name | Power state |
|----------------|------------|---|
| Emulated Off | OFF_EMU | <p>Emulated off mode permits you to debug the powerup and powerdown cycle without changing the software.</p> <p>In this mode, the core powerdown is normal, except:</p> <ul style="list-style-type: none"> • The clock is not gated and power is not removed when the core is powered down. • Only the Warm reset is asserted. The debug logic is preserved in the core and remains accessible by the debugger. |
| Debug recovery | DBG_RECOV | <p>The RAM and logic are powered up.</p> <p>This mode is for applying a Warm reset to the cluster, while preserving memory and RAS registers for debug purposes. Both cache and RAS state are preserved when transitioning from DBG_RECOV to ON.</p> <p>Caution: This mode must not be used during normal system operation.</p> |
| Warm reset | WARM_RST | A Warm reset resets all state except for the trace logic and the debug and RAS registers. |

Deviating from the legal power modes can lead to **UNPREDICTABLE** results. You must comply with the dynamic power management and powerup and powerdown sequences described in [5.6 Neoverse V2 core powerup and powerdown sequence](#) on page 47.

The following figure shows the supported modes for the Neoverse™ V2 core power domain and the legal transitions between them.

Figure 5-3: Neoverse™ V2 core power mode transitions**Related information**

[5.2 Architectural clock gating modes](#) on page 40

[5.2.1 Wait for Interrupt and Wait for Event](#) on page 40

[5.4.4 Full retention mode](#) on page 45

5.4.1 On mode

In On power mode, the Neoverse™ V2 core is on and fully operational.

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

5.4.2 Off mode

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

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

A Cold reset can reset the core in this mode.

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

5.4.3 Emulated off mode

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

5.4.4 Full retention mode

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

The core can enter into Full retention mode when all of the following conditions are met:

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

The core can exit Full retention mode when it detects any of the following:

- A WFI or WFE wakeup event, as defined in the [Arm® Architecture Reference Manual for A-profile architecture](#).
- An event that requires the core clock to be temporarily enabled without exiting the WFI or WFE low-power state. For example, an L1 or L2 snoop, a cache or TLB maintenance operation, a debug access on the debug APB bus, or a GIC access.

Related information

[5.2.1 Wait for Interrupt and Wait for Event](#) on page 40

5.4.5 Debug recovery mode

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

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

Debug recovery also supports preserving the *Reliability, Availability, and Serviceability* (RAS) state, in addition to the cache contents. In this case, a transition to debug recovery is made from any state.

When 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 On mode.



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

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

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.

5.4.6 Warm reset mode

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

A Warm reset is applied to the Neoverse™ V2 core when the core receives a Warm reset signal from the DSU-110 side of the CPU bridge:

The Neoverse™ V2 core implements the Arm®v8-A Reset Management Register, RMR_EL3. When running in EL3, setting the RMR_EL3.RR bit to 1 requests a Warm reset.

See the [Arm® Architecture Reference Manual for A-profile architecture](#) for more information about RMR_EL3.

5.5 Performance and power management

The Neoverse™ V2 core implements *Performance and Power Management* (PPM) features that can be used to limit high activity events within the core, or trade off efficiency versus peak performance.

The PPM features are:

- *Maximum Power Mitigation Mechanism* (MPMM)
- *Performance Defined Power* (PDP)

5.5.1 Maximum Power Mitigation Mechanism

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

If the count of high-activity events exceeds a pre-defined threshold during an evaluation period, MPMM temporarily limits the rate of instruction execution and memory system transactions.

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

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

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



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

5.5.2 Performance Defined Power

Performance Defined Power (PDP) is a power management feature that trades off peak performance for a reduced power envelope on general workloads.

The PDP is configured using a level of aggressiveness among three possible values. When the level of aggressiveness is increased, the average workload power is reduced but it causes more performance loss, which varies by workload.

The PDP has an impact on:

- Core power reduction. The core power is reduced and the efficiency is increased.
- External memory system power reduction. Memory request bandwidth is modulated to reduce power in the memory system.

5.6 Neoverse™ V2 core powerup and powerdown sequence

There is no specific sequence to power up the Neoverse™ V2 core. To power down the core, you must follow a specific sequence. There are no software steps required to bring a core into coherence after reset.

To powerdown the Neoverse™ V2 core:

1. If required, save the state of the core to system memory to allow for retrieval of the core state during core powerup.
2. Disable interrupts to the core.
 - a. Disable the interrupt enable bits in the ICC_IGRPEN0_EL1 and ICC_IGRPEN1_EL1 registers.
 - b. Set the GIC distributor wake-up request for the core using the GICR_WAKER register.
 - c. Read the GICR_WAKER register to confirm that the ChildrenAsleep bit indicates that the interface is quiescent.
3. Disable the interrupt outputs from the RAS registers. Alternatively, re-direct the core RAS fault and error interrupt outputs to the system error manager. For more information, see [5.6.1 Managing RAS fault and error interrupts during the core powerdown](#) on page 48.
4. Set the IMP_CPUPWRCTLR_EL1.CORE_PWRDN_EN bit to 1 to indicate to the power controller that a powerdown is requested.
5. Execute an `ISB` instruction.
6. Execute a `WFI` instruction. Once the `WFI` instruction is executed, the powerdown sequence cannot be interrupted.

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

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

When the IMP_CPUPWRCTLR_EL1.CORE_PWRDN_EN bit is set, executing a `WFI` instruction automatically masks all interrupts and wakeup events in the core. As a result, applying a reset is the only way to wake up the core from the *Wait for Interrupt* (WFI) state.

5.6.1 Managing RAS fault and error interrupts during the core powerdown

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

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

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

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

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

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

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

If preferable, you can disable the generation of RAS faults and error interrupts for correctable and deferrable errors while enabling the error interrupt for uncorrectable errors. However, the core error interrupt output must be re-routed to the system error manager before executing the WFI instruction in the core powerdown procedure. To do this, configure the ERxCTLR_EL1 register as follows:

- ERxCTLR_EL1.CFI = 0
- ERxCTLR_EL1.FI = 0
- ERxCTLR_EL1.UI = 1

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

5.7 Debug over powerdown

The Neoverse™ V2 core supports debug over powerdown, which allows a debugger to retain its connection with the core even when powered down. This behavior enables debug to continue

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

The debug over powerdown logic is part of the DebugBlock in the DSU-110. The DebugBlock is external to the cluster, and must remain powered on during the debug over powerdown process.

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

6. Memory management

The *Memory Management Unit* (MMU) is responsible for translating an input address to an output address. This translation is based on address mapping and memory attribute information that is available in the Neoverse™ V2 core internal registers and translation tables. The MMU also controls memory access permissions, memory ordering, and cache policies for each region of memory.

An address translation from an input address to an output address is described as a stage of address translation. The Neoverse™ V2 core can perform:

- Stage 1 translations that translate an input *Virtual Address* (VA) to an output *Physical Address* (PA) or *Intermediate Physical Address* (IPA).
- Stage 2 translations that translate an input IPA to an output PA.
- Combined stage 1 and stage 2 translations that translate an input VA to an IPA, and then translate that IPA to an output PA. The Neoverse™ V2 core performs translation table walks for each stage of the translation.

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

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

See the [Arm® Architecture Reference Manual for A-profile architecture](#) for more information on this feature.

6.1 Memory Management Unit components

The Neoverse™ V2 *Memory Management Unit* (MMU) includes several *Translation Lookaside Buffers* (TLBs), an *MMU Translation Cache* (MMUTC), and a translation table prefetcher.

A TLB is a cache of recently executed page translations within the MMU. The Neoverse™ V2 core implements a two-level TLB structure.

A TLB stores all page sizes and is responsible for breaking these down into smaller pages when required for the L1 data or instruction TLB.

The following table describes the MMU components.

Table 6-1: MMU components

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

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

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

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

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

A miss in the L1 data TLB and a hit in the L2 TLB has a 6-cycle penalty compared to a hit in the L1 data TLB. This penalty can be increased depending on the arbitration of pending requests.

6.2 Translation Lookaside Buffer entry content

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

Each TLB entry contains a *Virtual Address* (VA), a *Physical Address* (PA), and a set of memory properties that includes type and access permissions.

Each TLB entry is associated with either a particular *Address Space Identifier* (ASID) or a global indicator. Each TLB entry also contains a field to store the *Virtual Machine Identifier* (VMID) in the entry applicable to accesses from EL0 and EL1. The VMID permits hypervisor virtual machine switches without requiring the TLB to be invalidated.

Related information

[6.4 Translation table walks](#) on page 54

6.3 Translation Lookaside Buffer match process

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

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

- EL3 in Secure state
- EL2, or EL0 in *Virtualization Host Extensions* (VHE) mode, in secure state and 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 *Address Space Identifier* (ASID) matches the current ASID held in the TTBR0_ELx or TTBR1_ELx register associated with the target translation regime, or the entry is marked global.
- The *Virtual Machine Identifier* (VMID) matches the current VMID held in the VTTBR_EL2 register.

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

- EL2 in secure state and 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. It is also relevant in Secure state when SCR_EL3.EEL2 is 1.

6.4 Translation table walks

When the Neoverse™ V2 core generates a memory access, the *Memory Management Unit* (MMU) searches for the requested *Virtual Address* (VA) in the *Translation Lookaside Buffers* (TLBs). If it is not present, then it is a miss and the MMU proceeds by looking up the translation table during a translation table walk.

When the Neoverse™ V2 core generates a memory access, the MMU:

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

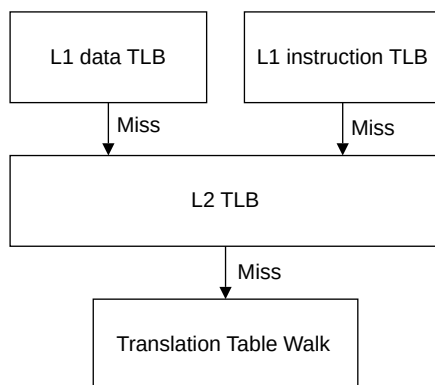
Address translation is performed only when the MMU is enabled. They can also be disabled for a particular translation base register, in which case the MMU returns a translation fault.

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

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

The following figure shows the translation table walk process.

Figure 6-1: Translation table walks



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

Related information

7. [L1 instruction memory system](#) on page 59

8. [L1 data memory system](#) on page 63

9. [L2 memory system](#) on page 68

6.5 Hardware management of the Access flag and dirty state

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

This feature is enabled in TCR_ELx and VTCR_EL2. Translation table descriptors include the *Dirty Bit Modifier* (DBM) field to support the hardware management of dirty state.

The Neoverse™ V2 core supports hardware updates to the Access flag and to dirty state only when the translation tables are held in Inner Write-Back and Outer Write-Back Normal memory regions. If software requests a hardware update in a region that is not Inner Write-Back or Outer Write-Back Normal memory, then the Neoverse™ V2 core returns an abort with the following encoding:

- ESR_ELx.DFSC = 0b110001 for Data Aborts
- ESR_ELx.IFSC = 0b110001 for Instruction Aborts

6.6 Responses

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

MMU responses

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

- An L1 instruction or data *Translation Lookaside Buffer* (TLB) hit
- An L2 TLB hit
- A translation table walk

The responses from the MMU contain the following information:

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

MMU aborts

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

External aborts

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

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

Neoverse™ V2 takes a synchronous abort on a Normal memory ldrx that receives a non-EXOK response from CHI. The abort is asynchronous for Device memory ldrx. For strx, OK and EXOK responses are expected and do not cause aborts. NDErr and DErr responses for WriteNoSnp Excl=1 cause asynchronous aborts.

Misprogramming contiguous hints

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

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

If there is this kind of mis-programming, then the Neoverse™ V2 core does not generate a translation fault.

Conflict aborts

The Neoverse™ V2 core does not generate Conflict aborts.

6.7 Memory behavior and supported memory types

The Neoverse™ V2 core supports memory types defined in the Armv8-A architecture.

Device memory types have the following attributes:

G – Gathering

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

R – Reordering

The capability to reorder transactions

E – Early Write Acknowledgement

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



In the following table, n denotes a "non-" prefix, that is, n means the capability is not allowed.

The following table shows how memory types are supported in the Neoverse™ V2 core.

Table 6-2: Supported device memory types

| Memory attribute type | Shareability | Inner Cacheability | Outer Cacheability | Notes |
|-----------------------|---|--|-----------------------------------|--|
| Device nGnRnE | Outer Shareable | - | - | Treated as Device nGnRnE |
| Device nGnRE | Outer Shareable ¹ | - | - | Treated as Device nGnRE |
| Device nGRE | Outer Shareable ¹ | - | - | Treated as Device nGRE |
| Device GRE | Outer Shareable ¹ | - | - | Treated as Device GRE |
| Normal | Outer Shareable ¹ | Non-cacheable | Any | Treated as Non-cacheable |
| Normal | Outer Shareable ¹ | Write-Through Cacheable | Any | Treated as Non-cacheable |
| Normal | Outer Shareable ¹ | Write-Back Cacheable | Non-cacheable | Treated as Non-cacheable |
| Normal | Outer Shareable ¹ | Write-Back Cacheable | Write-Through Cacheable | Treated as Non-cacheable |
| Normal | See Table 6-3: Shareability for Normal memory on page 57. | Write-Back Cacheable (any allocation hint) | Write-Back Cacheable No Allocate | Treated as Write-Back Read and Write Allocate but the outer cacheability propagated to the DSU-110 is 0 (No Allocate) |
| Normal | See Table 6-3: Shareability for Normal memory on page 57. | Write-Back Cacheable (any allocation hint) | Write-Back Read or Write Allocate | Treated as Write-Back Read and Write Allocate but the outer cacheability propagated to the DSU-110 is 1, therefore upgraded to Write and Read Allocate |

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

Table 6-3: Shareability for Normal memory

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

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

6.8 Page-based hardware attributes

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

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

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

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

PBHA combination between stage 1 and stage 2 on memory accesses

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

When stage 1 and stage 2 are enabled:

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

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

Mismatched aliases

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

7. L1 instruction memory system

The Neoverse™ V2 L1 memory system is responsible for fetching instructions and predicting branches. It includes the L1 instruction cache, the L1 instruction *Translation Lookaside Buffer* (TLB), and the *Macro-operation* (MOP) cache.

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

The following table shows the L1 instruction memory system features.

Table 7-1: L1 instruction memory system features

| Feature | Description |
|------------------------------------|---|
| L1 instruction cache | 64KB |
| | 4-way set associative |
| | <i>Virtually Indexed, Physically Tagged</i> (VIPT) behaving as <i>Physically Indexed, Physically Tagged</i> (PIPT) |
| | Always protected with parity |
| Cache line length | 64 bytes |
| <i>Macro-operation</i> (MOP) cache | 1536 macro-operations |
| | 4-way skewed associative |
| | <i>Virtually Indexed, Virtually Tagged</i> (VIVT) behaving as <i>Physically Indexed, Physically Tagged</i> (PIPT) |
| | Level 0 instruction cache working in the fetch stages of the pipeline to improve throughput and latency |
| Cache policy | L1 I-cache Pseudo- <i>Least Recently Used</i> (LRU) cache replacement policy for L1 L0 MOP-cache <i>Not Recently Used</i> (NRU) replacement policy |



Note

The L1 instruction TLB also resides in the L1 instruction memory system. However, it is part of the *Memory Management Unit* (MMU) and is described in [6. Memory management](#) on page 51.

7.1 L1 instruction cache behavior

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

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

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

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



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

Related information

[5.4.5 Debug recovery mode](#) on page 45

7.2 L1 instruction cache Speculative memory accesses

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

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

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

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

7.3 Program flow prediction

The Neoverse™ V2 core contains program flow prediction hardware, also known as branch prediction. Branch prediction increases overall performance and reduces power consumption.

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

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

Program flow prediction hardware contains the following functionality:

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

Predicted and non-predicted instructions

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

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

Exception return branch instructions are not predicted.

Return stack

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

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

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

Any of the following instructions cause a return stack pop:

- RET
- RETAA
- RETAB

The following instructions are not predicted:

- ERET
- ERETAA
- ERETAB

7.4 Instruction cache hardware coherency

When the optional instruction cache hardware coherency option is configured using the `COHERENT_ICACHE` parameter, the following behaviors in the core are affected:

- L1 instruction cache and L2 cache become strictly inclusive. Any cache line present in the L1 instruction cache is also present in the L2 cache.
- Instruction cache invalidate instructions are treated as no-ops and do not cause instruction cache invalidation or DVMMsg broadcasts to other cores.
- L2 cache monitors all store and cache invalidation coherency traffic and ensures that the L1 instruction cache invalidates any entry that is written to, or invalidated from, the L2 cache.
- `CTR_EL0[29]` reads as 1. Using this register, software can discover that the core implements instruction cache hardware coherency and can optimize functions to not issue instruction cache instructions.

The following restriction and recommendation applies to configuring instruction cache hardware coherency in the core:

- The coherency domain containing a core configured with instruction cache hardware coherency must not contain any coherent masters that require software instruction cache maintenance.
- Arm recommends systems consisting of a large number of Neoverse™ V2 cores should configure the cores with instruction cache coherency to eliminate possible performance issues related to instruction cache instruction broadcasts as DVMMsg transactions to all masters in the system.

8. L1 data memory system

The Neoverse™ V2 L1 data memory system is responsible for executing load and store instructions, as well as specific instructions such as atomics, cache maintenance operations, and memory tagging instructions. It includes the L1 data cache and the L1 data *Translation Lookaside Buffer* (TLB).

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

The following table shows the L1 data memory system features.

Table 8-1: L1 data memory system features

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



Note

The L1 data TLB also resides in the L1 instruction memory system. However, it is part of the *Memory Management Unit* (MMU) and is described in [6. Memory management](#) on page 51.

8.1 L1 data cache behavior

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

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

There is no operation to invalidate the entire data cache. If software requires this function, then it must be constructed by iterating over the cache geometry and executing a series of individual invalidates by set/way instructions. DCCISW operations perform both a clean and invalidate of the target set/way. The values of HCR_EL2.SWIO have no effect.

If the L1 data cache is disabled, then:

- A new line is not allocated in the L2 cache as a result of an instruction fetch
- All load and store instructions to cacheable memory are treated as Non-cacheable

- Data cache maintenance operations continue to execute normally

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

To maintain data coherency between multiple cores, the Neoverse™ V2 core uses the *Modified Exclusive Shared Invalid* (MESI) protocol.

Related information

[5.4.5 Debug recovery mode](#) on page 45

8.2 Instruction implementation in the L1 data memory system

The Neoverse™ V2 core supports the atomic instructions added in the Arm®v8.1-A architecture. Atomic instructions to Cacheable memory can be performed as either near atomics or far atomics, depending on where the cache line containing the data resides.

If an instruction hits in the L1 data cache, then the Neoverse™ V2 core tries to perform it as a near atomic. Then, based on system behavior, the core can decide to perform it as a far atomic.

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

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

The Neoverse™ V2 core supports atomics to Device or Non-cacheable memory, however this relies on the interconnect also supporting atomics. If such an atomic instruction is executed when the interconnect does not support them, then it results in an abort.

8.3 Internal exclusive monitor

The Neoverse™ V2 core includes an internal exclusive monitor with a 2-state, open and exclusive state machine that manages Load-Exclusive and Store-Exclusive accesses and Clear-Exclusive (`CLREX`) instructions.

You can use these instructions to construct semaphores, ensuring synchronization between different processes running on the core, and also between different cores that are using the same

coherent memory locations for the semaphore. A Load-Exclusive instruction tags a small block of memory for exclusive access. CTR_ELO defines the size of the tagged blocks as 16 words, one cache line.



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

See the [Arm® Architecture Reference Manual for A-profile architecture](#) for more information on these instructions.

8.4 Data prefetching

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

Preload instructions

The Neoverse™ V2 core supports the AArch64 prefetch memory instructions, `PRFM`.

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

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

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

For more information about prefetch memory and preloading caches, see the [Arm® Architecture Reference Manual for A-profile architecture](#).

Data prefetching and monitoring

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

The `IMP_CPUECTLR_EL1` register controls the prefetcher. For more information, see [A.1.15 IMP_CPUECTLR_EL1, CPU Extended Control Register](#) on page 167.

Data cache zero

In the Neoverse™ V2 core, the *Data Cache Zero by Virtual Address* (`DC ZVA`) instruction enables a block of 64 bytes in memory, aligned to 64 bytes in size, to be set to zero.

For more information, see the [Arm® Architecture Reference Manual for A-profile architecture](#).

8.5 Write streaming mode

The Neoverse™ V2 core supports write streaming mode, sometimes referred to as read allocate mode, both for the L1 and the L2 cache.

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

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

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



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

The memory system continues in write streaming mode until either:

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

When a Neoverse™ V2 core has switched to write streaming mode, the memory system continues to monitor the write traffic. It signals to the L2, System Level Cache, or DRAM, to go into write streaming mode when it observes a further number of full cache line writes.

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

- `IMP_CPUECTLR_EL1.WS_THR_L2` configures the L2 write streaming mode threshold.
- `IMP_CPUECTLR_EL1.WS_THR_L3` configures the SLC write streaming threshold.
- `IMP_CPUECTLR_EL1.WS_THR_L4` has no effect.
- `IMP_CPUECTLR_EL1.DRAM_WR_THR` configures the DRAM write streaming mode threshold.

Related information

[A.1.15 IMP_CPUECTLR_EL1, CPU Extended Control Register](#) on page 167

9. L2 memory system

The Neoverse™ V2 L2 memory system connects the core with the DSU-110 through the CPU bridge. It includes the L2 *Translation Lookaside Buffer* (TLB) and private L2 cache.

The L2 cache is unified and private to each Neoverse™ V2 core in a cluster.

The following table shows the L2 memory system features.

Table 9-1: L2 memory system features

| Feature | Type |
|----------------------------|--|
| L2 cache | 1MB or 2MB |
| | 8-way set associative, 4 banks |
| | <i>Physically Indexed, Physically Tagged</i> (PIPT) |
| | Always protected with <i>Error Correcting Code</i> (ECC) |
| Cache line length | 64 bytes |
| Cache policy | Dynamic biased cache replacement policy |
| Interface with the DSU-110 | One CHI Issue E compliant interface with 256-bit read and write DAT channel widths |

9.1 L2 cache

The integrated L2 cache handles both instruction and data requests from the instruction and data side in addition to translation table walk requests and snoops from the CHI interconnect.

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

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

Related information

[5.4.5 Debug recovery mode](#) on page 45

9.2 Support for memory types

The Neoverse™ V2 core simplifies coherency logic by downgrading some memory types.

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

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

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

The additional attribute hints are used as follows:

Allocation hint

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

Transient hint

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

Transient lines evicted from the L2 cache do not allocate downstream caches.

9.3 Transaction capabilities

The CHI Issue E interface between the Neoverse™ V2 L2 memory system and the DSU-110 provides transaction capabilities for the core.

The following table shows the maximum possible values for read, write, *Distributed Virtual Memory* (DVM) issuing, and snoop capabilities of the Neoverse™ V2 L2 cache. There is a maximum limit of 92 outstanding transactions overall.

Table 9-2: Neoverse™ V2 transaction capabilities

| Attribute | Maximum value | Description |
|-----------------------------|---------------|---|
| Write issuing capability | 92 | This is the maximum number of outstanding write transactions. |
| Read issuing capability | 92 | This is the maximum number of outstanding read transactions. |
| Snoop acceptance capability | 53 | This is the maximum number of outstanding snoops accepted. |
| DVM issuing capability | 92 | This is the maximum number of outstanding DVM operation transactions. |

10. Direct access to internal memory

The Neoverse™ V2 core provides a mechanism to read the internal memory that the L1 and L2 caches and *Translation Lookaside Buffer* (TLB) structures use through **IMPLEMENTATION DEFINED** system registers. This functionality can be useful when investigating issues where the coherency between the data in the cache and data in system memory is broken.



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

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

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

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

10.1 L1 cache encodings

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

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

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

Table 10-2: Neoverse™ V2 L1 instruction cache tag location encoding

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

| Bit field of Xn | Description |
|-----------------|------------------------|
| [17:14] | Reserved |
| [13:6] | Virtual address [13:6] |
| [5:0] | Reserved |

Table 10-3: Neoverse™ V2 L1 instruction cache data location encoding

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

Table 10-4: Neoverse™ V2 L1 instruction TLB data location encoding

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

Table 10-5: Neoverse™ V2 L0 Macro-operation cache data location encoding

| Bit field of Xn | Description |
|-----------------|--------------|
| [31:24] | RAMID = 0x06 |
| [23:10] | Reserved |
| [9:0] | Index [9:0] |

Table 10-6: Neoverse™ V2 L1 data cache tag location encoding

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

| Bit field of Xn | Description |
|-----------------|-------------|
| [5:0] | Reserved |

Table 10-7: Neoverse™ V2 L1 data cache data location encoding

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

Table 10-8: Neoverse™ V2 L1 data TLB location encoding

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

10.1.1 L1 instruction tag RAM returned data

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

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

Table 10-9: L1 instruction cache tag format for Instruction Register 0

| Bit field | Description |
|-----------|--|
| [63:40] | Reserved |
| [39] | Non-secure identifier for the physical address |
| [38:3] | Physical address [47:12] |
| [2:1] | Instruction state [1:0] 0b00 Invalid 0b01 <i>Reserved for Future Use</i> (RFU) 0b10 RFU 0b11 Valid |
| [0] | Parity |

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

| Bit field | Description |
|-----------|-------------|
| [63:0] | Reserved |

Table 10-11: L1 instruction cache tag format for Instruction Register 2

| Bit field | Description |
|-----------|-------------|
| [63:0] | Reserved |

10.1.2 L1 instruction data RAM returned data

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

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

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

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

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

| Bit field | Description |
|-----------|--------------|
| [63:20] | 0 |
| [19:0] | Data [83:64] |

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

| Bit field | Description |
|-----------|-------------|
| [63:0] | 0 |

10.1.3 L1 instruction TLB returned data

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

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

Table 10-15: L1 instruction TLB format for Instruction Register 0

| Bit field | Description |
|-----------|----------------------|
| [63] | Virtual address [12] |
| [62:59] | PBHA [3:0] |
| [58] | TLB attribute |

| Bit field | Description |
|-----------|---|
| [57:55] | <p>Memory attributes:</p> <p>0b000 Device nGnRnE</p> <p>0b001 Device nGnRE</p> <p>0b010 Device nGRE</p> <p>0b011 Device GRE</p> <p>0b100 Non-cacheable</p> <p>0b101 Write-Back No-Allocate</p> <p>0b110 Write-Back Transient</p> <p>0b111 Write-Back Read-Allocate and Write-Allocate</p> |
| [54:52] | <p>Page size:</p> <p>0b000 4KB</p> <p>0b001 16KB</p> <p>0b010 64KB</p> <p>0b100 2MB</p> <p>Other Reserved</p> |
| [51] | Outer-shared |
| [50] | Inner-shared |
| [49:42] | 0 |
| [41:40] | TLB attribute |
| [39:24] | ASID[15:0] |
| [23:8] | VMID[15:0] |

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

Table 10-16: L1 instruction TLB format for Instruction Register 1

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

Table 10-17: L1 instruction TLB format for Instruction Register 2

| Bit field | Description |
|-----------|--------------------------|
| [63:11] | Reserved |
| [10:9] | TLB attribute |
| [8] | Non-Secure |
| [7:0] | Physical address [47:40] |

10.1.4 L0 macro-operation RAM returned data

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

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

Table 10-18: L0 MOP cache format for Instruction Register 0

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

Table 10-19: L0 MOP cache format for Instruction Register 1

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

Table 10-20: L0 MOP cache format for Instruction Register 2

| Bit field | Description |
|-----------|-------------|
| [63:0] | 0 |

10.1.5 L1 data tag RAM returned data

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

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

Table 10-21: L1 data cache tag format for Data Register 0

| Bit field | Description |
|-----------|---|
| [63:28] | Physical address [47:12] |
| [27:25] | Reserved |
| [24] | Transient/WBNA |
| [23:20] | <i>Memory Tagging Extension</i> (MTE) tag poison |
| [19:4] | MTE tag data |
| [3:2] | MTE tag state: 0b00 Invalid 0b01 Shared 0b11 Dirty state |
| [1:0] | MESI: 0b00 Invalid 0b01 Shared 0b10 Exclusive 0b11 Modified |

Table 10-22: L1 data cache tag format for Data Register 1

| Bit field | Description |
|-----------|-----------------------|
| [0] | Non-secure identifier |
| [8:1] | ECC |

Table 10-23: L1 data cache tag format for Data Register 2

| Bit field | Description |
|-----------|-------------|
| [63:0] | 0 |

10.1.6 L1 data data RAM returned data

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

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

Table 10-24: L1 data cache data format for Data Register 0

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

Table 10-25: L1 data cache data format for Data Register 1

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

Table 10-26: L1 data cache data format for Data Register 2

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

10.1.7 L1 data TLB returned data

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

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

Table 10-27: L1 data TLB format for Data Register 0

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

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

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

Table 10-28: L1 data TLB format for Data Register 1

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

Table 10-29: L1 data TLB format for Data Register 2

| Bit field | Description |
|-----------|-------------------------|
| [13] | Tagged MTE |
| [12] | FWB override |
| [11] | PBHA [3] |
| [10] | PBHA [2] |
| [9] | PBHA [1] |
| [8] | PBHA [0] |
| [7:0] | Physical address[47:40] |

10.2 L2 cache encodings

The L2 cache is 8-way set associative.

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

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

Table 10-30: Neoverse™ V2 L2 cache tag location encoding for 1MB²

| Bit field of Xn | Description |
|-----------------|--------------|
| [31:24] | RAMID = 0x10 |
| [23:21] | Reserved |
| [20:19] | Way (0-7) |
| [18:17] | Reserved |
| [16:12] | Index[16:12] |

² Index[16:8]=XOR(physical address[16:8], physical address[25:17]), Index[7:6]=XOR(physical address[7:6], physical address[11:10])

| Bit field of Xn | Description |
|-----------------|---|
| [11:9] | XOR(Index[11:9], Way[2:0]) |
| [8] | Index[8] |
| [7:6] | XOR(Index[7:6], Index[11:10], Way[2:1]) |
| [5:0] | Reserved |

Table 10-31: Neoverse™ V2 L2 cache tag location encoding for 2MB³

| Bit field of Xn | Description |
|-----------------|----------------------------------|
| [31:24] | RAMID = 0x10 |
| [23:22] | Reserved |
| [21:19] | Way (0-7) |
| [18] | Reserved |
| [17:11] | Index[17:11] |
| [10:8] | XOR(Index[10:8], Way[2:0]) |
| [7] | XOR(Index[7], Index[11]) |
| [6] | XOR(Index[6], Index[10], Way[2]) |
| [5:0] | Reserved |

Table 10-32: Neoverse™ V2 L2 cache data location encoding for 1MB

| Bit field of Xn | Description |
|-----------------|---|
| [31:24] | RAMID = 0x11 |
| [23:22] | Reserved |
| [21:19] | Way (0-7) |
| [18:17] | Reserved |
| [16:12] | XOR(Index[16:12], 5'b01000) |
| [11:9] | XOR(Index[11:9], Way[2:0]) |
| [8] | Index[8] |
| [7:6] | XOR(Index[7:6], Index[11:10], Way[2:1]) |
| [5:4] | Physical address[5:4] |
| [3:0] | Reserved |

Table 10-33: Neoverse™ V2 L2 cache data location encoding for 2MB

| Bit field of Xn | Description |
|-----------------|-------------------------------|
| [31:24] | RAMID = 0x11 |
| [23:20] | Reserved |
| [21:19] | Way (0-7) |
| [18] | Reserved |
| [17:11] | XOR(Index[17:11], 7'b0001000) |
| [10:8] | XOR(Index[10:8], Way[2:0]) |

³ Index[17:8]=XOR(physical address[17:8], physical address[27:18]), Index[7:6]=XOR(physical address[7:6], physical address[11:10])

| Bit field of Xn | Description |
|-----------------|----------------------------------|
| [7] | XOR(Index[7], Index[11]) |
| [6] | XOR(Index[6], Index[10], Way[2]) |
| [5:4] | Physical address[5:4] |
| [3:0] | Reserved |

Table 10-34: Neoverse™ V2 L2 TLB location encoding

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

Table 10-35: Neoverse™ V2 L2 victim location encoding

| Bit field of Rd | Description |
|-----------------|-------------------------------|
| [31:24] | RAMID = 0x12 |
| [23:18] | Reserved |
| [17:16] | Index[17:16] |
| [15] | INV(Index[15]) |
| [14:8] | Index[14:8] |
| [7:6] | XOR(Index[7:6], Index[11:10]) |
| [5:0] | Reserved |

10.2.1 L2 tag RAM returned data

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

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

For 1MB L2 cache without coherent instruction cache

n=41, m=17

For 1MB L2 cache with coherent instruction cache

n=42, m=17

For 2MB L2 cache without coherent instruction cache

n=40, m=18

For 2MB L2 cache with coherent instruction cache

n=41, m=18

Table 10-36: L2 tag cache format for Data Register 0 without coherent instruction cache

| Bit field | Description |
|-------------|-------------|
| [n+22:n+16] | ECC |

| Bit field | Description |
|------------|---|
| [n+15] | MPAM_PMG |
| [n+14:n+6] | MPAM_PARTID[8:0] |
| [n+5] | MPAM_NS |
| [n+4:n+1] | PBHA[3:0] |
| [n:11] | Physical tag [47:m] |
| [10] | Non-secure |
| [9] | Reserved |
| [8:7] | Virtual address [13:12] |
| [6] | Shareable |
| [5] | L1 data cache valid |
| [4:3] | MTE state: 0b00 Invalid 0b10 Clean 0b11 Dirty |
| [2:0] | L2 state: 0b101 UniqueDirty 0b001 UniqueClean 0bx11 SharedClean 0bxx0 Invalid |

Table 10-37: L2 tag cache format for Data Register 0 with coherent instruction cache

| Bit field | Description |
|-------------|-------------------------|
| [63:n+20] | ECC |
| [n+19] | MPAM_PMG |
| [n+18:n+10] | MPAM_PARTID[8:0] |
| [n+9] | MPAM_NS |
| [n:8:n+5] | Instruction cache valid |
| [n+4:n+1] | PBHA[3:0] |
| [n:12] | Physical tag [47:m] |
| [11] | Non-secure |
| [10] | Reserved |
| [9:8] | Virtual address [13:12] |
| [7] | Shareable |
| [6] | L1 data cache shared |

| Bit field | Description |
|-----------|---|
| [5] | L1 data cache valid |
| [4:3] | MTE state: 0b00 Invalid 0b10 Clean 0b11 Dirty |
| [2:0] | L2 state: 0b101 UniqueDirty 0b001 UniqueClean 0bx11 SharedClean 0bxx0 Invalid |

Table 10-38: L2 tag cache format for Data Register 1 without coherent instruction cache

| Bit field | Description |
|-----------|-------------|
| [63:0] | 0 |

Table 10-39: L2 tag cache format for Data Register 1 with coherent instruction cache

| Bit field | Description |
|-----------|-------------|
| [63:n-36] | 0 |
| [n-37:0] | ECC |

Table 10-40: L2 tag cache format for Data Register 2

| Bit field | Description |
|-----------|-------------|
| [63:0] | 0 |

10.2.2 L2 data RAM returned data

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

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

Table 10-41: L2 data RAM format for Data Register 0

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

Table 10-42: L2 data RAM format for Data Register 1

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

Table 10-43: L2 data RAM format for Data Register 2

| Bit field | Description |
|-----------|-----------------------|
| [63:20] | 0 |
| [19:12] | ECC for Data [127:64] |
| [11:4] | ECC for Data [63:0] |
| [3:0] | MTE tags |

10.2.3 L2 TLB RAM returned data

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

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

Table 10-44: L2 TLB format for Instruction Register 0

| Bit field | Description |
|-----------|---|
| [63:62] | Reserved |
| [61:20] | Physical address When bit[6] is 0: <ul style="list-style-type: none"> [61:26] = PA[47:12] [25:20] = Reserved When bit[6] is 1: <ul style="list-style-type: none"> [61:28] = PA[47:14] [27:26] = PA[13:12] for page 3 (highest memory address) [25:24] = PA[13:12] for page 2 [23:22] = PA[13:12] for page 1 [21:20] = PA[13:12] for page 0 (lowest memory address) |

| Bit field | Description |
|-----------|--|
| [19:17] | Page size: 0b000 4KB 0b001 16KB 0b010 64KB 0b100 2MB 0b101 32MB 0b110 512MB 0b111 1GB |
| [16:7] | Reserved |
| [6] | Coalesced entry |
| [5:2] | Valid bits |
| [1:0] | Reserved |

Table 10-45: L2 TLB format for Instruction Register 1

| Bit field | Description |
|-----------|---------------------------------|
| [63:59] | ASID[4:0] |
| [58:55] | PBHA |
| [54] | Walk cache entry |
| [53:25] | Virtual address[48:20] |
| [24:21] | Reserved |
| [20] | Non-secure |
| [19:9] | Reserved |
| [8] | nG, indicates a non global page |
| [7] | Outer Shareable |
| [6] | Inner Shareable |
| [5] | Outer allocate |

| Bit field | Description |
|-----------|--|
| [4:2] | Memory attributes: 0b000 Device nGnRnE 0b001 Device nGnRE 0b010 Device nGRE 0b011 Device GRE 0b100 Non-cacheable 0b101 Write-Back No-Allocate 0b110 Write-Back Transient 0b111 Write-Back Read-Allocate and Write-Allocate |
| [1:0] | Reserved |

Table 10-46: L2 TLB format for Instruction Register 2

| Bit field | Description |
|-----------|---|
| [63:30] | Reserved |
| [29:27] | MSID[2:0]: 0b000 Secure EL1 0b001 Secure EL2 0b010 Non-secure EL1 0b011 Non-secure EL2 0b101 EL3 |
| [26:11] | VMID[15:0] |
| [10:0] | ASID[15:5] |

10.2.4 L2 Victim RAM returned data

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

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

Table 10-47: Neoverse™ V2 L2 victim format for data register 0

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

Table 10-48: Neoverse™ V2 L2 victim format for data register 1

| Bit field of Rd | Description |
|-----------------|-------------|
| [63:0] | 0 |

Table 10-49: Neoverse™ V2 L2 victim format for data register 2

| Bit field of Rd | Description |
|-----------------|-------------|
| [63:0] | 0 |

11. RAS Extension support

The Neoverse™ V2 core supports the *Reliability, Availability, and Serviceability* (RAS) Extension, including all extensions up to Arm®v9.0-A.

In particular, the Neoverse™ V2 core supports:

- Cache protection with *Single Error Correct Double Error Detect* (SECCDED) ECC on the RAMs that contain dirty data. This includes the L1 data cache tag and data RAMs, the L2 cache tag and data RAMs, and the L2 *Transaction Queue* (TQ) RAMs.
- Cache protection with *Single Error Detect* (SED) parity on the RAMs that only contain clean data. This includes the L1 instruction cache tag and data cache, the *Macro-operation* (MOP) cache, and the *Memory Management Unit* (MMU) RAMs.
- The *Error Synchronization Barrier* (ESB) instruction. When an ESB instruction is executed, the core ensures that all SError Interrupts that are generated by instructions before the ESB are either taken by the core or pended in DISR_EL1.
- Poison attribute on bus transfers
- Error Data Record registers
- *Fault Handling Interrupts* (FHIs)
- *Error Recovery Interrupts* (ERIs)
- Error injection

The Neoverse™ V2 core features the following node:

- Node 0 that includes the private L1 and L2 memory systems in the core

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

11.1 Cache protection behavior

The configuration of the *Reliability, Availability, and Serviceability* (RAS) Extension that is implemented in the Neoverse™ V2 core includes cache protection. In this case, the Neoverse™ V2 core protects against errors that result in a RAM bitcell holding the incorrect value.

The RAMs in the Neoverse™ V2 core have the following capability:

SEC parity

Single Error Correct. One bit of parity is applicable to the entire word. Errors are corrected by refetching cached data.

SECCDED ECC

Single Error Correct, Double Error Detect. The word size is specific for each RAM and depends on the protection granule.

The following table shows which protection type is applied to each RAM in the Neoverse™ V2 core. The core can progress and remain functionally correct when there is a single bit error in any RAM.

Table 11-1: RAM cache protection

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

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

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

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

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

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

11.2 Error containment

The Neoverse™ V2 core supports error containment for data errors, which means that detected errors are not silently propagated.

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

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

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

11.3 Fault detection and reporting

When the Neoverse™ V2 core detects a fault, it raises a *Fault Handling Interrupt* (FHI) exception or an *Error Recovery Interrupt* (ERI) exception through the fault or the error signals. FHIs and ERIs are reflected in the *Reliability, Availability, and Serviceability* (RAS) registers, which are updated in the node that detects the errors.

Fault handling interrupts

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

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

Error recovery interrupts

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

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

Related information

[A.11 AArch64 RAS register summary](#) on page 388

[A.11.4 ERXCTLR_EL1, Selected Error Record Control Register](#) on page 394

[A.11.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register](#) on page 397

11.4 Error detection and reporting

When the Neoverse™ V2 core consumes an error, it raises different exceptions depending on the error type.

The Neoverse™ V2 core might raise:

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

11.4.1 Error reporting and performance monitoring

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

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

In Secure state, the event is counted only if MDCR_EL3.SPME is asserted. See the [Arm® Architecture Reference Manual for A-profile architecture](#) for a description of MDCR_EL3.

Related information

[18.1 Performance monitors events](#) on page 110

11.5 Error injection

Error injection consists of inserting an error in the error detection logic to verify the reporting and recording structure.

Error injection uses the error detection and reporting registers to insert errors. The Neoverse™ V2 core can inject the following error types:

Corrected errors

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

Deferred errors

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

Uncontainable errors

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

An error can be injected immediately or when a 32-bit counter reaches zero. You can control the value of the counter through the ERXPFGCDN register. The value of the counter decrements on a per clock cycle basis.

For more information on the ERXPFGCDN register, see the *Arm® Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile*.



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

11.6 AArch64 RAS registers

The summary table provides an overview of all **IMPLEMENTATION DEFINED** RAS registers in the core. For more information about a register, you can click the register name in the table.

Table 11-2: AArch64 RAS register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-------------------------------|-----|-----|-----|-----|-----|---------------------------|--------|---|
| ERRIDR_EL1 | 3 | C5 | 0 | C3 | 0 | See individual bit resets | 64-bit | Error Record ID Register |
| ERRSELR_EL1 | 3 | C5 | 0 | C3 | 1 | See individual bit resets | 64-bit | Error Record Select Register |
| ERXFR_EL1 | 3 | C5 | 0 | C4 | 0 | See individual bit resets | 64-bit | Selected Error Record Feature Register |
| ERXCTLR_EL1 | 3 | C5 | 0 | C4 | 1 | 0x0 | 64-bit | Selected Error Record Control Register |
| ERXSTATUS_EL1 | 3 | C5 | 0 | C4 | 2 | 0x0 | 64-bit | Selected Error Record Primary Status Register |
| ERXADDR_EL1 | 3 | C5 | 0 | C4 | 3 | See individual bit resets | 64-bit | Selected Error Record Address Register |
| ERXPFGF_EL1 | 3 | C5 | 0 | C4 | 4 | See individual bit resets | 64-bit | Selected Pseudo-fault Generation Feature register |
| ERXPFGCTL_EL1 | 3 | C5 | 0 | C4 | 5 | 0x1000 | 64-bit | Selected Pseudo-fault Generation Control register |
| ERXPFGCDN_EL1 | 3 | C5 | 0 | C4 | 6 | See individual bit resets | 64-bit | Selected Pseudo-fault Generation Countdown register |
| ERXMISCO_EL1 | 3 | C5 | 0 | C5 | 0 | See individual bit resets | 64-bit | Selected Error Record Miscellaneous Register 0 |
| ERXMISC1_EL1 | 3 | C5 | 0 | C5 | 1 | 0x0 | 64-bit | Selected Error Record Miscellaneous Register 1 |
| ERXMISC2_EL1 | 3 | C5 | 0 | C5 | 2 | 0x0 | 64-bit | Selected Error Record Miscellaneous Register 2 |
| ERXMISC3_EL1 | 3 | C5 | 0 | C5 | 3 | 0x0 | 64-bit | Selected Error Record Miscellaneous Register 3 |

12. GIC CPU interface

The *Generic Interrupt Controller* (GIC) supports and controls interrupts. The GIC connects to the Neoverse™ V2 core through a GIC CPU interface. The GIC CPU interface includes registers to mask, identify, and control the state of interrupts that are forwarded to the core.

Each core has a GIC CPU interface, which connects to a common external distributor component.

The GICv4.1 architecture implemented in the Neoverse™ V2 core supports:

- Two security states
- *Secure Virtualization Extension* (SVE)
- *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 either Group 0 or Group 1, where Group 0 interrupts are always Secure
- Signaling Group 1 interrupts to the target core using either the IRQ or the FIQ exception request. Group 1 interrupts can be Secure or Non-secure
- Signaling Group 0 interrupts to the target core using the FIQ exception request only
- A unified scheme for handling the priority of Group 0 and Group 1 interrupts

See the [Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4](#) for more information about interrupt groups.

12.1 Disable the GIC CPU interface

The Neoverse™ V2 core always includes the *Generic Interrupt Controller* (GIC) CPU interface, however you can disable it to meet your requirements.

To disable the GIC CPU interface, assert the GICCDISABLE signal HIGH at reset. If you disable it this way, then you can use other GIC architectures to drive the nFIQ and nIRQ interrupt signals. If the Neoverse™ V2 core is not integrated with an external GIC interrupt distributor component in the system (minimum GICv3 architecture), then you need to disable the GIC CPU interface.

If you disable the GIC CPU interface, then:

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



Note

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

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

12.2 AArch64 GIC registers

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Generic Interrupt Controller (GIC) registers in the core. For more information about a register, you can click the register name in the table.

Table 12-1: AArch64 GIC register summary

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

13. Advanced SIMD and floating-point support

The Neoverse™ V2 core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set without floating-point exception trapping. The Neoverse™ V2 core floating-point implementation includes Arm®v8.3-A and Arm®v8.5-A features.

The Neoverse™ V2 core implements all scalar operations in hardware with support for all combinations of:

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

14. Scalable Vector Extensions support

The Neoverse™ V2 core supports the *Scalable Vector Extension* (SVE) and the *Scalable Vector Extension 2* (SVE2). SVE and SVE2 complement and do not replace AArch64 Advanced SIMD and floating-point functionality.

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

The Neoverse™ V2 core implements a scalable vector length of 128 bits.

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

15. System control

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

The main functions of the system registers are:

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

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

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

15.1 AArch64 Generic system control registers

The summary table provides an overview of all **IMPLEMENTATION DEFINED** generic system control registers in the core. For more information about a register, you can click the register name in the table.

Table 15-1: AArch64 Generic system control register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|----------------------------------|-----|-----|-----|-----|-----|---------------------------|--------|---|
| AIDR_EL1 | 3 | C0 | 1 | C0 | 7 | 0x0 | 64-bit | Auxiliary ID Register |
| ACTLR_EL1 | 3 | C1 | 0 | C0 | 1 | 0x0 | 64-bit | Auxiliary Control Register (EL1) |
| ACTLR_EL2 | 3 | C1 | 4 | C0 | 1 | 0x0 | 64-bit | Auxiliary Control Register (EL2) |
| HACR_EL2 | 3 | C1 | 4 | C1 | 7 | 0x0 | 64-bit | Hypervisor Auxiliary Control Register |
| ACTLR_EL3 | 3 | C1 | 6 | C0 | 1 | 0x0 | 64-bit | Auxiliary Control Register (EL3) |
| AMAIR_EL2 | 3 | C10 | 0 | C3 | 0 | 0x0 | 64-bit | Auxiliary Memory Attribute Indirection Register (EL2) |
| LORID_EL1 | 3 | C10 | 0 | C4 | 7 | See individual bit resets | 64-bit | LORegionID (EL1) |
| AMAIR_EL1 | 3 | C10 | 5 | C3 | 0 | 0x0 | 64-bit | Auxiliary Memory Attribute Indirection Register (EL1) |
| AMAIR_EL3 | 3 | C10 | 6 | C3 | 0 | 0x0 | 64-bit | Auxiliary Memory Attribute Indirection Register (EL3) |
| RMR_EL3 | 3 | C12 | 6 | C0 | 2 | See individual bit resets | 64-bit | Reset Management Register (EL3) |
| IMP_CPUACTLR_EL1 | 3 | C15 | 0 | C1 | 0 | See individual bit resets | 64-bit | CPU Auxiliary Control Register (EL1) |

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-------------------------|-----|-----|-----|-----|-----|---------------------------|--------|---|
| IMP_CPUACTLR2_EL1 | 3 | C15 | 0 | C1 | 1 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 2 (EL1) |
| IMP_CPUACTLR3_EL1 | 3 | C15 | 0 | C1 | 2 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 3 (EL1) |
| IMP_CPUACTLR4_EL1 | 3 | C15 | 0 | C1 | 3 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 4 (EL1) |
| IMP_CPUECTLR_EL1 | 3 | C15 | 0 | C1 | 4 | See individual bit resets | 64-bit | CPU Extended Control Register |
| IMP_CPUECTLR2_EL1 | 3 | C15 | 0 | C1 | 5 | See individual bit resets | 64-bit | CPU Extended Control Register 2 |
| IMP_CPUBUSQOS_EL1 | 3 | C15 | 0 | C1 | 7 | See individual bit resets | 64-bit | CPU Bus QoS Register |
| IMP_CPUPPMCR3_EL3 | 3 | C15 | 0 | C2 | 4 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUPPMPCR_EL1 | 3 | C15 | 0 | C2 | 4 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUL2DIRTYLNCT_EL1 | 3 | C15 | 0 | C2 | 5 | 0x0 | 64-bit | CPU L2 Dirty Line Count Register |
| IMP_CUPWRCTLR_EL1 | 3 | C15 | 0 | C2 | 7 | 0x0 | 64-bit | CPU Power Control Register |
| IMP_ATCR_EL1 | 3 | C15 | 0 | C7 | 0 | 0x0 | 64-bit | CPU Auxiliary Translation Control Register (EL1) |
| IMP_CPUACTLR5_EL1 | 3 | C15 | 0 | C8 | 0 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 5 (EL1) |
| IMP_CPUACTLR6_EL1 | 3 | C15 | 0 | C8 | 1 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 6 (EL1) |
| IMP_CPUACTLR7_EL1 | 3 | C15 | 0 | C8 | 2 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 7 (EL1) |
| IMP_ATCR_EL2 | 3 | C15 | 4 | C7 | 0 | 0x0 | 64-bit | CPU Auxiliary Translation Control Register (EL2) |
| IMP_AVTCR_EL2 | 3 | C15 | 4 | C7 | 1 | 0x0 | 64-bit | CPU Virtualization Auxiliary Translation Control Register (EL2) |
| IMP_CPUPPMCR_EL3 | 3 | C15 | 6 | C2 | 0 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUPMCCR_EL3 | 3 | C15 | 6 | C2 | 1 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUPPMCR2_EL3 | 3 | C15 | 6 | C2 | 1 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUL2SDIRTYLNCT_EL3 | 3 | C15 | 6 | C2 | 3 | 0x0 | 64-bit | CPU L2 Secure Dirty Line Count Register |
| IMP_CUPDPTUNE_EL3 | 3 | C15 | 6 | C2 | 4 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUPPMCR4_EL3 | 3 | C15 | 6 | C2 | 4 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CUPDPTUNE2_EL3 | 3 | C15 | 6 | C2 | 5 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUPPMCR5_EL3 | 3 | C15 | 6 | C2 | 5 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUPMCRMTUNE_EL3 | 3 | C15 | 6 | C2 | 6 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-------------------|-----|-----|-----|-----|-----|---------------------------|--------|---|
| IMP_CPUPPMCR6_EL3 | 3 | C15 | 6 | C2 | 6 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUACTLR_EL3 | 3 | C15 | 6 | C4 | 0 | See individual bit resets | 64-bit | CPU Auxiliary Control Register (EL3) |
| IMP_ATCR_EL3 | 3 | C15 | 6 | C7 | 0 | 0x0 | 64-bit | CPU Auxiliary Translation Control Register (EL2) |
| IMP_CPUPSELR_EL3 | 3 | C15 | 6 | C8 | 0 | See individual bit resets | 64-bit | Selected Instruction Private Select Register |
| IMP_CPUPCR_EL3 | 3 | C15 | 6 | C8 | 1 | See individual bit resets | 64-bit | Selected Instruction Private Control Register |
| IMP_CPUPOR_EL3 | 3 | C15 | 6 | C8 | 2 | See individual bit resets | 64-bit | Selected Instruction Private Opcode Register |
| IMP_CPUPMR_EL3 | 3 | C15 | 6 | C8 | 3 | See individual bit resets | 64-bit | Selected Instruction Private Mask Register |
| IMP_CPUPOR2_EL3 | 3 | C15 | 6 | C8 | 4 | See individual bit resets | 64-bit | Selected Instruction Private Opcode Register 2 |
| IMP_CPUPMR2_EL3 | 3 | C15 | 6 | C8 | 5 | See individual bit resets | 64-bit | Selected Instruction Private Mask Register 2 |
| IMP_CPUPFR_EL3 | 3 | C15 | 6 | C8 | 6 | See individual bit resets | 64-bit | Selected Instruction Private Flag Register |
| FPCR | 3 | C4 | 3 | C4 | 0 | See individual bit resets | 64-bit | Floating-point Control Register |
| AFSR0_EL2 | 3 | C5 | 0 | C1 | 0 | 0x0 | 64-bit | Auxiliary Fault Status Register 0 (EL2) |
| AFSR1_EL2 | 3 | C5 | 0 | C1 | 1 | 0x0 | 64-bit | Auxiliary Fault Status Register 1 (EL2) |
| AFSR0_EL1 | 3 | C5 | 5 | C1 | 0 | 0x0 | 64-bit | Auxiliary Fault Status Register 0 (EL1) |
| AFSR1_EL1 | 3 | C5 | 5 | C1 | 1 | 0x0 | 64-bit | Auxiliary Fault Status Register 1 (EL1) |
| AFSR0_EL3 | 3 | C5 | 6 | C1 | 0 | 0x0 | 64-bit | Auxiliary Fault Status Register 0 (EL3) |
| AFSR1_EL3 | 3 | C5 | 6 | C1 | 1 | 0x0 | 64-bit | Auxiliary Fault Status Register 1 (EL3) |

16. Random number generator support

The Neoverse™ V2 core can be configured to support two random number instructions introduced in the Arm®v8.5-A extension.

The following instructions return a 64-bit random number into a general purpose register.

- MRS Xn, RNDR
- MRS Xn, RNDRRS

The Neoverse™ V2 core expects the *True Random Number Generator* (TRNG) and the *Deterministic Random Bit Generator* (DRBG) to be available as a memory-mapped peripheral and must be capable of the following requirements.

- Design the TRNG and DRBG as architecturally stipulated.
- Provide as many copies of TRNG and DRBG as is necessary to meet the overall bandwidth and latency requirements of the system.
- Reseed the DRBG from the TRNG when a RNDRRS instruction is received, as defined by the address encoding described in the Neoverse™ V2 core microarchitecture.
- Provide *Quality of Service* (QOS) managed access to DRBG bandwidth as architecturally defined.
- Provide access to each TRNG and DRBG block through a memory-mapped Dev-nGnRnE read (LDP Xreg).
- The address used by the LDP Xreg for RNDR and RNDRRS instructions is a physical-address defined as follows:
 - A combination of base register of 64K page (CPURNDBR_EL3[47:16]), PE-specific identifier (CPURNDPEID_EL3[10:0]), instruction-type.
 - RNDR address: {CPURNDBR_EL3[47:16], CPURNDPEID_EL3[10:0], 1'b0, 4'b0}
 - RNDRRS address: {CPURNDBR_EL3[47:16], CPURNDPEID_EL3[10:0], 1'b1, 4'b0}
- Set CPURNDBR_EL3[47:16] in each Neoverse™ V2 core to match the peripheral base of the TRNG and DRBG block corresponding to the core. The association of the core to TRNG and DRBG block is defined by the system integrator.
- The TRNG and DRBG block must correctly decode the read-address, using [15:5] as the unique core identifier for QOS guarantees.
- The TRNG and DRBG block must correctly decode bit [4] of the read-address, 0 specifying an RNDR instruction and 1 specifying an RNDRRS instruction.
- Upon receiving a RNDR or RNDRRS request, the TRNG and DRBG block must return a 64-bit random number in the first 64 bits and 1 in the second 64 bits. In the event that the DRBG block is unable to provide a random number within a system integrator defined timeframe, it will return 0 in the first and second 64 bits.
- In the event of a bus error, a RNDR or RNDRRS request will fail and the core will set the PSTATE.Z flag and assert a SEI.



Note

The random number generator can be tested by running the *National Institute of Standards and Technology* (NIST) tests available as part of the *SBSA Architecture Compliance Suite* (ACS). The SBSA ACS is available at <https://github.com/ARM-software/sbsa-acs>. The NIST tests are available at https://github.com/ARM-software/sbsa-acs/tree/master/test_pool/nist_sts

16.1 AArch64 Random number control registers

The summary table provides an overview of all **IMPLEMENTATION DEFINED** random number control registers in the core. For more information about a register, you can click the register name in the table.

Table 16-1: AArch64 Random number control register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|------------------------------------|-----|-----|-----|-----|-----|-------|--------|--|
| IMP_CPURNDBR_EL3 | 3 | C15 | 6 | C3 | 0 | 0x0 | 64-bit | CPU Random Number Base Register |
| IMP_CPURNDPEID_EL3 | 3 | C15 | 6 | C3 | 1 | 0x0 | 64-bit | CPU Random Number Packet Identification Register |

17. Debug

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

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

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

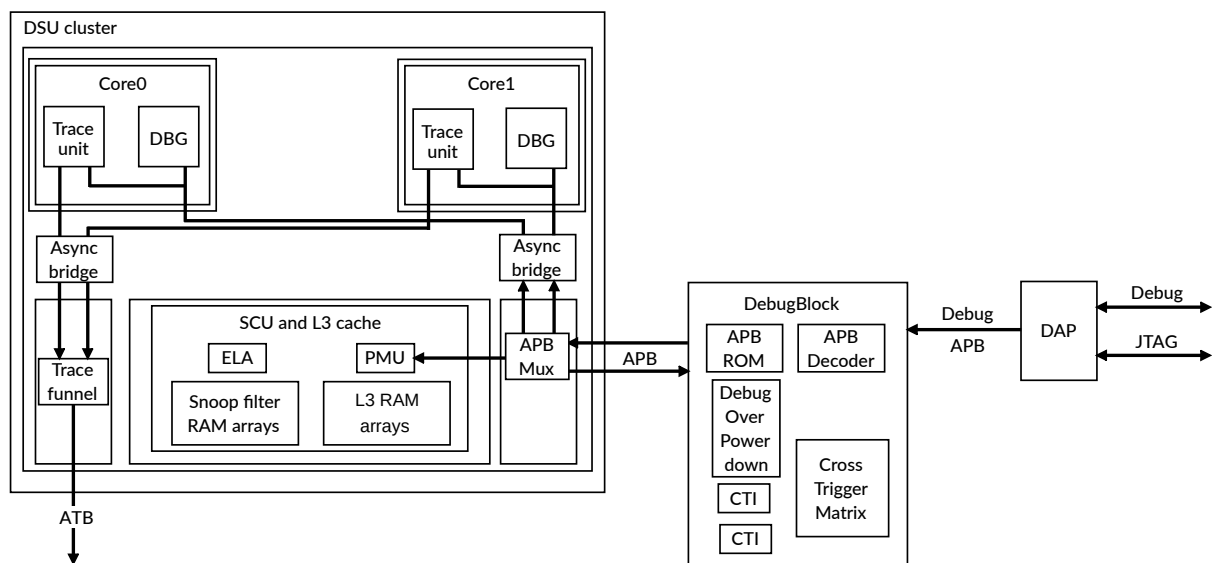
The connection between the cluster and the DebugBlock consists of a pair of *Advanced Peripheral Bus* APB interfaces, one in each direction. All debug traffic, except the authentication interface, takes place over this interface as read or write APB transactions. This debug traffic includes register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

The debug system implements the following CoreSight debug components:

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

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

Figure 17-1: DSU cluster debug components



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

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

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

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

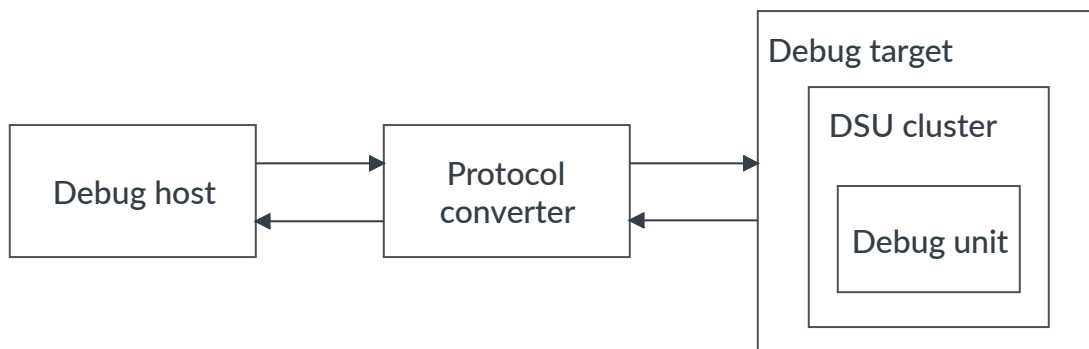
The Neoverse™ V2 core also supports direct access to internal memory, that is, cache debug. Direct access to internal memory allows software to read the internal memory that the L1 and L2 cache and *Translation Lookaside Buffer* (TLB) structures use. See [10. Direct access to internal memory](#) on page 70 for more information.

17.1 Supported debug methods

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

The following figure shows a typical external debug system.

Figure 17-2: External debug system



Debug host

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

Protocol converter

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

Debug target

The lowest level of the system implements system support for the protocol converter to access the debug unit. For DSU-110 based devices, the mechanism used to access the debug unit is based on the CoreSight architecture. The DSU-110 DebugBlock is accessed using an APB interface and the debug accesses are then directed to the selected V2 core inside the DSU cluster. An example of a debug target is a development system with a test chip or a silicon part with a V2 core.

Debug unit

Helps debugging software that is running on the core:

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

With the debug unit, you can:

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

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

17.2 Debug register interfaces

The Neoverse™ V2 core implements the Arm®v9.0-A Debug architecture. It also supports the Arm®v8.4-A Debug architecture and the Arm®v8.3-A Debug over powerdown.

The Debug architecture defines a set of Debug registers. The Debug register interfaces provide access to these registers either from software running on the core or from an external debugger.

Related information

[5.7 Debug over powerdown](#) on page 49

17.2.1 Core interfaces

System register access allows the Neoverse™ V2 core to access certain Debug registers directly. The Debug register interfaces provide access to these registers either from software running on the core or from an external debugger.

Access to the Debug registers is partitioned as follows:

Debug

This function is both system register based and memory-mapped. You can access the Debug register map using the APB slave port that connects into the DebugBlock of the DSU-110.

Performance monitoring

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

Trace

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

Statistical profiling

This function is system register based.

ELA registers

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

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

17.2.2 Effects of resets on debug registers

The complexporeset_n and complexreset_n signals of the core affect the debug registers.

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

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

17.2.3 External access permissions to Debug registers

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

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

Table 17-1: External access conditions to registers

| Name | Condition | Description |
|---------|-------------------------------------|---|
| Off | EDPRSR.PU = 1 | Because Armv8.3-DoPD, Debug over PowerDown, is implemented, access to this field is <i>Read-As-One</i> (RAO). When the core power domain is in a powerup state, the Debug registers in the core power domain can be accessed. When the core power domain is off, accesses to the Debug registers in the core power domain, including EDPRSR, return an error. |
| OSLK | OSLSR_EL1.OSLK = 1 | OS Lock is locked. |
| EDAD | AllowExternalDebugAccess() == FALSE | External debug access is disabled. If an error is returned because of an EDAD condition code, and this is the highest priority error condition, then EDPRSR.SDAD is set to 1. If not, then SDAD is unchanged. |
| Default | - | This is normal access, none of the conditions apply. |

17.2.4 Breakpoints and watchpoints

The Neoverse™ V2 core supports six breakpoints, four watchpoints, and a standard *Debug Communications Channel* (DCC).

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

17.3 Debug events

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

The Neoverse™ V2 core responds to a debug event in one of the following ways:

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

In the Neoverse™ V2 core, watchpoint debug events are always synchronous. Memory hint instructions and cache clean operations, except `DC ZVA`, and `DC IVAC` do not generate watchpoint debug events. Store exclusive instructions generate a watchpoint debug event even when the check for the control of exclusive monitor fails. Atomic `CAS` instructions generate a watchpoint debug event even when the compare operation fails.

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

17.4 Debug memory map and debug signals

The debug memory map and debug signals are handled at cluster level.

See the *Debug* chapter of the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*.

17.5 ROM table

The Neoverse™ V2 core includes a ROM table that contains a list of components in the system. Debuggers can use the ROM table to determine which CoreSight components are implemented.

The ROM table is a CoreSight debug related component that aids system debug along with CoreSight SoC and is for the Neoverse™ V2 core. There is one ROM table for each core and ROM tables comply with the *Arm® CoreSight™ Architecture Specification v3.0*.

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

The Neoverse™ V2 core ROM table includes the following entries:

Table 17-2: Core ROM table

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

Related information

[B.1 External CoreROM register summary](#) on page 437

17.6 CoreSight component identification

Each component associated with the Neoverse™ V2 core has a unique set of CoreSight ID values.

Table 17-3: Neoverse™ V2 CoreSight component identification

| Component | Peripheral ID | Component ID | DevType | DevArch | Revision |
|------------|---------------|--------------|------------|------------|----------|
| Trace unit | 0x04100BBD4F | 0xB105900D | 0x00000013 | 0x47705A13 | rOp1 |
| PMU | 0x04100BBD4F | 0xB105900D | 0x00000016 | 0x47702A16 | rOp1 |
| DBG | 0x04100BBD4F | 0xB105900D | 0x00000015 | 0x47709A15 | rOp1 |
| ROM Table | 0x04100BBD4F | 0xB105900D | 0x00000000 | 0x47700AF7 | rOp1 |

For details on the CoreSight component identification for the Neoverse™ V2 core ELA, see the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual*.

17.7 AArch64 Debug registers

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Debug registers in the core. For more information about a register, you can click the register name in the table.

Table 17-4: AArch64 Debug register summary

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

17.8 External Debug registers

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

Table 17-5: External Debug register summary

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

| Offset | Name | Reset | Width | Description |
|--------|-------------------------|---------------------------|--------|--|
| 0xFF0 | EDCIDR0 | See individual bit resets | 32-bit | External Debug Component Identification Register 0 |
| 0xFF4 | EDCIDR1 | See individual bit resets | 32-bit | External Debug Component Identification Register 1 |
| 0xFF8 | EDCIDR2 | See individual bit resets | 32-bit | External Debug Component Identification Register 2 |
| 0xFFC | EDCIDR3 | See individual bit resets | 32-bit | External Debug Component Identification Register 3 |

17.9 External CoreROM registers

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

Table 17-6: External CoreROM register summary

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

18. Performance Monitors Extension support

The Neoverse™ V2 core implements the Performance Monitors Extension, including Arm®v8.4-A and Arm®v8.5-A performance monitoring features.

The Neoverse™ V2 core *Performance Monitoring Unit* (PMU):

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

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

18.1 Performance monitors events

The Neoverse™ V2 core *Performance Monitoring Unit* (PMU) collects events from other units in the design and uses numbers to reference these events.

The following table lists the Neoverse™ V2 core performance monitors events. Event reference numbers that are not listed are reserved.



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

Table 18-1: Performance monitors Events

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

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

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

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

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

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

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

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

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

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

| Event number | Event mnemonic | Event description |
|--------------|---------------------|--|
| 0x80BC | SVE_LDFF_SPEC | SVE First-fault load operations speculatively executed |
| 0x80BD | SVE_LDFF_FAULT_SPEC | SVE First-fault load operations speculatively executed which set FFR bit to 0 |
| 0x80C0 | FP_SCALE_OPS_SPEC | Scalable floating-point element operations speculatively executed |
| 0x80C1 | FP_FIXED_OPS_SPEC | Non-scalable floating-point element operations speculatively executed |
| 0x80E3 | ASE_SVE_INT8_SPEC | Operation counted by ASE_SVE_INT_SPEC where the largest type is 8-bit integer |
| 0x80E7 | ASE_SVE_INT16_SPEC | Operation counted by ASE_SVE_INT_SPEC where the largest type is 16-bit integer |
| 0x80EB | ASE_SVE_INT32_SPEC | Operation counted by ASE_SVE_INT_SPEC where the largest type is 32-bit integer |
| 0x80EF | ASE_SVE_INT64_SPEC | Operation counted by ASE_SVE_INT_SPEC where the largest type is 64-bit integer |

18.2 Performance monitors interrupts

Performance monitors interrupts indicate events that have been observed several times.

When the *Performance Monitoring Unit* (PMU) generates an interrupt, the nPMUIRQ[n] output is driven LOW.

See the *Performance Monitors Extension support* chapter of the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for more information.

18.3 External register access permissions

The Neoverse™ V2 core supports access to the *Performance Monitoring Unit* (PMU) registers from the system register interface and a memory-mapped interface.

Access to a register depends on:

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

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

18.4 AArch64 Performance monitors registers

The summary table provides an overview of all **IMPLEMENTATION DEFINED** performance monitors registers in the core. For more information about a register, you can click the register name in the table.

Table 18-2: AArch64 Performance monitors register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-----------------------------|-----|-----|-----|-----|-----|---------------------------|--------|---|
| PMMIR_EL1 | 3 | C9 | 0 | C14 | 6 | See individual bit resets | 64-bit | Performance Monitors Machine Identification Register |
| PMCR_ELO | 3 | C9 | 3 | C12 | 0 | See individual bit resets | 64-bit | Performance Monitors Control Register |
| PMCEID0_ELO | 3 | C9 | 3 | C12 | 6 | See individual bit resets | 64-bit | Performance Monitors Common Event Identification register 0 |
| PMCEID1_ELO | 3 | C9 | 3 | C12 | 7 | See individual bit resets | 64-bit | Performance Monitors Common Event Identification register 1 |

18.5 External PMU registers

The summary table provides an overview of all memory-mapped performance monitors registers in the core. Individual register descriptions provide detailed information.

Table 18-3: External PMU register summary

| Offset | Name | Reset | Width | Description |
|--------|---------------------------|---------------------------|--------|---|
| 0x600 | PMPCSSR | See individual bit resets | 64-bit | Snapshot Program Counter Sample Register |
| 0x608 | PMCIDSSR | See individual bit resets | 32-bit | Snapshot CONTEXTIDR_EL1 Sample Register |
| 0x60C | PMCID2SSR | See individual bit resets | 32-bit | Snapshot CONTEXTIDR_EL2 Sample Register |
| 0x610 | PMSSSR | See individual bit resets | 32-bit | PMU Snapshot Status Register |
| 0x618 | PMCCNTSR | See individual bit resets | 64-bit | PMU Cycle Counter Snapshot Register |
| 0x620 | PMEVCNTR0 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x628 | PMEVCNTR1 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x630 | PMEVCNTR2 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x638 | PMEVCNTR3 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x640 | PMEVCNTR4 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x648 | PMEVCNTR5 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x6F0 | PMSSCR | See individual bit resets | 32-bit | PMU Snapshot Capture Register |
| 0xE00 | PMCFGR | See individual bit resets | 32-bit | Performance Monitors Configuration Register |
| 0xE04 | PMCR_ELO | See individual bit resets | 32-bit | Performance Monitors Control Register |
| 0xE20 | PMCEID0 | See individual bit resets | 32-bit | Performance Monitors Common Event Identification register 0 |
| 0xE24 | PMCEID1 | See individual bit resets | 32-bit | Performance Monitors Common Event Identification register 1 |
| 0xE28 | PMCEID2 | See individual bit resets | 32-bit | Performance Monitors Common Event Identification register 2 |
| 0xE2C | PMCEID3 | See individual bit resets | 32-bit | Performance Monitors Common Event Identification register 3 |

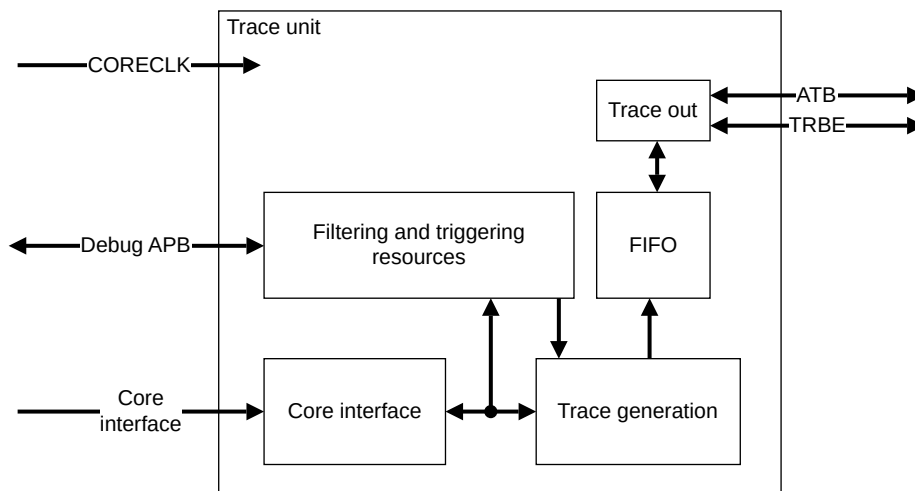
| Offset | Name | Reset | Width | Description |
|--------|-----------|---------------------------|--------|---|
| 0xE40 | PMMIR | See individual bit resets | 32-bit | Performance Monitors Machine Identification Register |
| 0xFBC | PMDEVARCH | See individual bit resets | 32-bit | Performance Monitors Device Architecture register |
| 0xFC8 | PMDEVID | See individual bit resets | 32-bit | Performance Monitors Device ID register |
| 0xFCC | PMDEVTYPE | See individual bit resets | 32-bit | Performance Monitors Device Type register |
| 0xFD0 | PMPIDR4 | See individual bit resets | 32-bit | Performance Monitors Peripheral Identification Register 4 |
| 0xFE0 | PMPIDR0 | See individual bit resets | 32-bit | Performance Monitors Peripheral Identification Register 0 |
| 0xFE4 | PMPIDR1 | See individual bit resets | 32-bit | Performance Monitors Peripheral Identification Register 1 |
| 0xFE8 | PMPIDR2 | See individual bit resets | 32-bit | Performance Monitors Peripheral Identification Register 2 |
| 0xFEC | PMPIDR3 | See individual bit resets | 32-bit | Performance Monitors Peripheral Identification Register 3 |
| 0xFF0 | PMCIDR0 | See individual bit resets | 32-bit | Performance Monitors Component Identification Register 0 |
| 0xFF4 | PMCIDR1 | See individual bit resets | 32-bit | Performance Monitors Component Identification Register 1 |
| 0xFF8 | PMCIDR2 | See individual bit resets | 32-bit | Performance Monitors Component Identification Register 2 |
| 0xFFC | PMCIDR3 | See individual bit resets | 32-bit | Performance Monitors Component Identification Register 3 |

19. Embedded Trace Extension support

The Neoverse™ V2 core implements the *Embedded Trace Extension* (ETE). The trace unit performs real-time instruction flow tracing based on the ETE. The trace unit is a CoreSight component and is an integral part of the Arm Real-time Debug solution, the Arm Debugger. The Arm Debugger is a part of the Arm Development Studio.

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

Figure 19-1: Trace unit components



Core interface

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

Trace generation

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

Filtering and triggering resources

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

FIFO

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

Trace out

Trace from the FIFO is output on the AMBA ATB interface or to the trace buffer implemented by the *TRace Buffer Extension (TRBE)*.

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

19.1 Trace unit resources

Trace resources include counters, external inputs and outputs, and comparators.

The following table shows the trace unit resources, and indicates which of these resources Neoverse™ V2 core implements.

Table 19-1: Trace unit resources implemented

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

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

19.2 Trace unit generation options

The Neoverse™ V2 core trace unit implements a set of generation options.

The following table shows the trace generation options that are implemented in the Neoverse™ V2 core trace unit.

Table 19-2: Trace unit generation options implemented

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

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

19.3 Reset the trace unit

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

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

19.4 Program and read the trace unit registers

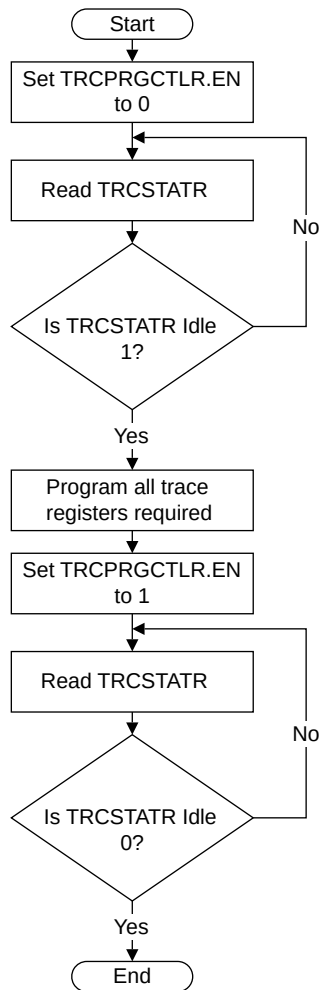
You program and read the trace unit registers using either the Debug APB interface or the System register interface.

The core does not have to be in debug state when you program the trace unit registers. When you program the trace unit registers, you must enable all the changes at the same time. Otherwise, if you program the counter, it might start to count based on incorrect events before the correct setup is in place for the trigger condition. To disable the trace unit, use the TRCPRGCTLR.EN bit.

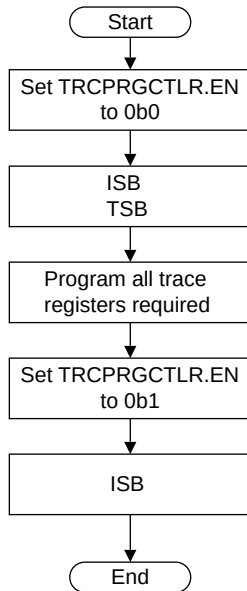
For more information on the following registers, see the *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile*:

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

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

Figure 19-2: Programming trace unit registers using the Debug APB interface

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

Figure 19-3: Programming trace unit registers using the System register interface

19.5 Trace unit register interfaces

The Neoverse™ V2 core supports an APB memory-mapped interface and a system register interface to trace unit registers.

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

19.6 Interaction with the Performance Monitoring Unit and Debug

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

Interaction with the PMU

The Neoverse™ V2 core includes a PMU that enables events, such as cache misses and executed instructions, to be counted over time.

The PMU and trace unit function together.

Use of PMU events by the trace unit

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

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

Related information

[18. Performance Monitors Extension support](#) on page 110

[18.1 Performance monitors events](#) on page 110

19.7 ETE events

The Neoverse™ V2 core trace unit collects events from other units in the design and uses numbers to reference these events.

Other than the events mentioned in [18.1 Performance monitors events](#) on page 110, the following events are also exported:

Table 19-3: ETE events

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

19.8 AArch64 Trace registers

The summary table provides an overview of all **IMPLEMENTATION DEFINED** trace registers in the core. For more information about a register, you can click the register name in the table.

Table 19-4: AArch64 Trace register summary

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

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

19.9 External ETE registers

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

Table 19-5: External ETE register summary

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

| Offset | Name | Reset | Width | Description |
|--------|--------------------------|---------------------------|--------|--------------------------------------|
| 0xFE8 | TRCPIDR2 | See individual bit resets | 32-bit | Peripheral Identification Register 2 |
| 0xFEC | TRCPIDR3 | See individual bit resets | 32-bit | Peripheral Identification Register 3 |
| 0xFF0 | TRCCIDR0 | See individual bit resets | 32-bit | Component Identification Register 0 |
| 0xFF4 | TRCCIDR1 | See individual bit resets | 32-bit | Component Identification Register 1 |
| 0xFF8 | TRCCIDR2 | See individual bit resets | 32-bit | Component Identification Register 2 |
| 0xFFC | TRCCIDR3 | See individual bit resets | 32-bit | Component Identification Register 3 |

20. Trace Buffer Extension support

The Neoverse™ V2 core implements the *TRace Buffer Extension* (TRBE). The TRBE writes the program flow trace generated by the trace unit directly to memory. The TRBE is programmed through System registers.

When enabled, the TRBE can:

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

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

20.1 Program and read the trace buffer registers

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

The core does not have to be in debug state when you program the TRBE registers. When you program the TRBE registers, you must enable all the changes at the same time. Otherwise, if you program the counter, it might start to count based on incorrect events before the correct setup is in place for the trigger condition. To disable the TRBE, use the TRBLIMITR_EL1.E bit.

20.2 Trace buffer register interface

The Neoverse™ V2 core supports a system register interface to *TRace Buffer Extension* (TRBE) registers.

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

21. Activity Monitors Extension support

The Neoverse™ V2 core implements the Activity Monitors Extension to the Arm®v8.4-A architecture. Activity monitoring has features similar to performance monitoring features, but is intended for system management use whereas performance monitoring is aimed at user and debug applications.

The activity monitors provide useful information for system power management and persistent monitoring. The activity monitors are read-only in operation and their configuration is limited to the highest Exception level implemented.

The Neoverse™ V2 core implements seven counters in two groups, each of which is a 64-bit counter that counts a fixed event. Group 0 has four counters 0-3, and Group 1 has three counters 10-12.

21.1 Activity monitors access

The Neoverse™ V2 core supports access to activity monitors from the system register interface and supports read-only memory-mapped access using the utility bus interface.

See the [Arm® Architecture Reference Manual for A-profile architecture](#) for information on the memory mapping of these registers.

Access enable bit

The access enable bit AMUSERENR_ELO.EN controls access from ELO to the activity monitors system registers.

The CPTR_EL2.TAM bit controls access from ELO and EL1 to the activity monitors system registers. The CPTR_EL3.TAM bit controls access from ELO, EL1, and EL2 to the Activity Monitors Extension system registers. The AMUSERENR_ELO.EN bit is configurable at EL1, EL2, and EL3. All other controls, as well as the value of the counters, are configurable only at the highest implemented Exception level. For a detailed description of access controls for the registers, see the [Arm® Architecture Reference Manual for A-profile architecture](#).

System register access

The activity monitors can be accessed using the MRS and MSR instructions.

External memory-mapped access

Activity monitors can be memory-mapped accessed from the utility bus interface. In this case, the activity monitors registers only provide read access to the Activity Monitor Event Counter Registers.

Base address for *Activity Monitoring Unit* (AMU) registers on the utility bus interface is 0x<n>90000 where “n” is the Neoverse™ V2 core instance number in the DSU-110 cluster.

These registers are treated as RAZ/WI if either:

- The register is marked as Reserved.
- The register is accessed in the wrong Security state.

21.2 Activity monitors counters

The Neoverse™ V2 core implements four activity monitors counters, 0-3, and three auxiliary counters, 10-12.

Each counter has the following characteristics:

- All events are counted in 64-bit wrapping counters that overflow when they wrap. There is no support for overflow status indication or interrupts.
- Any change in clock frequency, including when a `WFI` and `WFE` instruction stops the clock, can affect any counter.
- Events 0-3 and auxiliary events 10-12 are fixed, and the `AMEVTYPER0<n>_ELO` and `AMEVTYPER1<n>_ELO` evtCount bits are read-only.
- The activity monitor counters are reset to zero on a Cold reset of the power domain of the core. When the core is not in reset, activity monitoring is available.

21.3 Activity monitors events

Activity monitors events in the Neoverse™ V2 core are either fixed or programmable, and they map to the activity monitors counters.

The following table shows the mapping of counters to fixed events.

Table 21-1: Mapping of counters to fixed events

| Activity monitor counter <n> | Event | Event number | Description |
|------------------------------|----------------------|--------------|---|
| AMEVCNTR00 | CPU_CYCLES | 0x0011 | Core frequency cycles |
| AMEVCNTR01 | CNT_CYCLES | 0x4004 | Constant frequency cycles |
| AMEVCNTR02 | Instructions retired | 0x0008 | Instruction architecturally executed This counter increments for every instruction that is executed architecturally, including instructions that fail their condition code check. |
| AMEVCNTR03 | STALL_BACKEND_MEM | 0x4005 | Memory stall cycles This counter counts cycles in which the core is unable to dispatch instructions from the front end to the back end due to a back end stall caused by a miss in the last level of cache within the core clock domain. |
| AMEVCNTR10 | MPMM_THRESHOLD_GEAR0 | 0x0300 | Maximum Power Mitigation System (MPMM) Gear 0 activity period threshold exceeded |
| AMEVCNTR11 | MPMM_THRESHOLD_GEAR1 | 0x0301 | Maximum Power Mitigation System (MPMM) Gear 1 activity period threshold exceeded |

| Activity monitor counter <n> | Event | Event number | Description |
|------------------------------|----------------------|--------------|--|
| AMEVCNTR12 | MPMM_THRESHOLD_GEAR2 | 0x0302 | Maximum Power Mitigation System (MPMM) Gear 2 activity period threshold exceeded |

21.4 AArch64 Activity monitors registers

The summary table provides an overview of all **IMPLEMENTATION DEFINED** activity monitors registers in the core. For more information about a register, you can click the register name in the table.

Table 21-2: AArch64 Activity monitors register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|---------------------------------|-----|-----|-----|-----|-----|---------------------------|--------|--|
| AMEVTYPER10_ELO | 3 | C13 | 3 | C14 | 0 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 1 |
| AMEVTYPER11_ELO | 3 | C13 | 3 | C14 | 1 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 1 |
| AMEVTYPER12_ELO | 3 | C13 | 3 | C14 | 2 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 1 |
| AMCFGR_ELO | 3 | C13 | 3 | C2 | 1 | See individual bit resets | 64-bit | Activity Monitors Configuration Register |
| AMCGCR_ELO | 3 | C13 | 3 | C2 | 2 | See individual bit resets | 64-bit | Activity Monitors Counter Group Configuration Register |
| AMEVTYPER00_ELO | 3 | C13 | 3 | C6 | 0 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 0 |
| AMEVTYPER01_ELO | 3 | C13 | 3 | C6 | 1 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 0 |
| AMEVTYPER02_ELO | 3 | C13 | 3 | C6 | 2 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 0 |
| AMEVTYPER03_ELO | 3 | C13 | 3 | C6 | 3 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 0 |

21.5 External AMU registers

The summary table provides an overview of all memory-mapped activity monitors registers in the core. Individual register descriptions provide detailed information.

Table 21-3: External AMU register summary

| Offset | Name | Reset | Width | Description |
|--------|-----------------------------|---------------------------|--------|--|
| 0x400 | AMEVTYPER00 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 0 |
| 0x404 | AMEVTYPER01 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 0 |
| 0x408 | AMEVTYPER02 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 0 |
| 0x40C | AMEVTYPER03 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 0 |

| Offset | Name | Reset | Width | Description |
|--------|-----------------------------|---------------------------|--------|--|
| 0x480 | AMEVTYPER10 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 1 |
| 0x484 | AMEVTYPER11 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 1 |
| 0x488 | AMEVTYPER12 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 1 |
| 0x48C | AMEVTYPER13 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 1 |
| 0xCE0 | AMCGCR | See individual bit resets | 32-bit | Activity Monitors Counter Group Configuration Register |
| 0xE00 | AMCFGR | See individual bit resets | 32-bit | Activity Monitors Configuration Register |
| 0xE08 | AMIIDR | See individual bit resets | 32-bit | Activity Monitors Implementation Identification Register |
| 0xFBC | AMDEVARCH | See individual bit resets | 32-bit | Activity Monitors Device Architecture Register |
| 0xFCC | AMDEVTYPE | See individual bit resets | 32-bit | Activity Monitors Device Type Register |
| 0xFD0 | AMPIDR4 | See individual bit resets | 32-bit | Activity Monitors Peripheral Identification Register 4 |
| 0xFE0 | AMPIDR0 | See individual bit resets | 32-bit | Activity Monitors Peripheral Identification Register 0 |
| 0xFE4 | AMPIDR1 | See individual bit resets | 32-bit | Activity Monitors Peripheral Identification Register 1 |
| 0xFE8 | AMPIDR2 | See individual bit resets | 32-bit | Activity Monitors Peripheral Identification Register 2 |
| 0xFEC | AMPIDR3 | See individual bit resets | 32-bit | Activity Monitors Peripheral Identification Register 3 |
| 0xFF0 | AMCIDR0 | See individual bit resets | 32-bit | Activity Monitors Component Identification Register 0 |
| 0xFF4 | AMCIDR1 | See individual bit resets | 32-bit | Activity Monitors Component Identification Register 1 |
| 0xFF8 | AMCIDR2 | See individual bit resets | 32-bit | Activity Monitors Component Identification Register 2 |
| 0xFFC | AMCIDR3 | See individual bit resets | 32-bit | Activity Monitors Component Identification Register 3 |

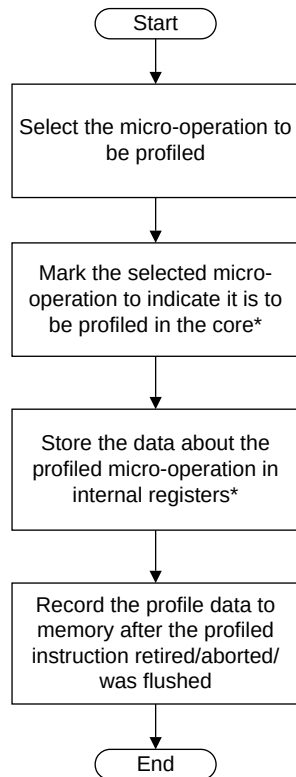
22. Statistical Profiling Extension support

The Neoverse™ V2 core implements the optional *Statistical Profiling Extension* (SPE) to the Arm®v8.5-A architecture. The SPE provides a statistical view of the performance characteristics of executed instructions that software writers can use to optimize their code for better performance.

The Neoverse™ V2 core profiles micro-operations to minimize the amount of logic necessary to support the SPE.

The following figure shows the SPE behavior in the Neoverse™ V2 core.

Figure 22-1: SPE behavior



* Throughout the lifetime of the micro-operation in the core

Profiles are collected periodically and a down-counter drives the selection of the micro-operations to be profiled. This counter counts the number of speculative micro-operations that are dispatched, decremented once for each micro-operation. When the counter reaches zero, a micro-operation is identified as being sampled and is profiled throughout its lifetime in the core.

SPE profiles are written to memory using a *Virtual Address* (VA), which means that writes of profiles must have access to the *Memory Management Unit* (MMU) to translate a VA to a *Physical Address* (PA), and must have a means to be written to memory.



Note

Profiling is expected to be largely non-intrusive to the performance of the core. The performance of the core is not meaningfully perturbed while profiling is taking place. The rate of occurrence depends on the sampling rate. You can specify a sampling rate that is meaningfully intrusive to the performance of the core. Arm recommends that the minimum sampling interval is once per 1024 micro-operations. This value is communicated to software through PMSIDR_EL1.Interval, bits[11:8].

See the [Arm® Architecture Reference Manual for A-profile architecture](#) for more information.

22.1 Statistical Profiling Extension events packet

The events packet indicates the **IMPLEMENTATION DEFINED** events that the sampled operation generated.

The following table shows the events defined in the 32-bit events packet implemented in the Neoverse™ V2 core.

Table 22-1: SPE events packet

| Bits | Definition |
|---------|---|
| [31:19] | Reserved |
| [18] | Empty predicate |
| [17] | Partial predicate |
| [16:13] | Reserved |
| [12] | Late prefetch |
| [11] | Data alignment flag |
| [10] | Remote access |
| [9] | Last level cache miss |
| [8] | Last level cache access |
| [7] | Branch mispredicted |
| [6] | Not taken |
| [5] | L1 data cache <i>Translation Lookaside Buffer</i> (TLB) |
| [4] | TLB access |
| [3] | L1 data cache refill |
| [2] | L1 data cache access |
| [1] | Architecturally retired |
| [0] | Generated exception |

22.2 Statistical Profiling Extension data source packet

The data source packet indicates where the data returned for a load or store operation was sourced.

The following table shows the data source defined in the 8-bit data source packet implemented in the Neoverse™ V2 core.

Table 22-2: SPE data source packet

| Value | Name |
|--------|-------------------------------------|
| 0b0000 | L1 data cache |
| 0b1000 | L2 cache |
| 0b1001 | Peer core |
| 0b1010 | Local cluster |
| 0b1011 | System cache |
| 0b1100 | Peer cluster |
| 0b1101 | Remote |
| 0b1110 | Dynamic Random Access Memory (DRAM) |

22.3 AArch64 Statistical Profiling Extension registers

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Statistical Profiling Extension registers in the core. For more information about a register, you can click the register name in the table.

Table 22-3: AArch64 Statistical Profiling Extension register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-----------------------------|-----|-----|-----|-----|-----|---------------------------|--------|--------------------------------|
| PMBIDR_EL1 | 3 | C9 | 0 | C10 | 7 | See individual bit resets | 64-bit | Profiling Buffer ID Register |
| PMSEVFR_EL1 | 3 | C9 | 0 | C9 | 5 | See individual bit resets | 64-bit | Sampling Event Filter Register |
| PMSIDR_EL1 | 3 | C9 | 0 | C9 | 7 | See individual bit resets | 64-bit | Sampling Profiling ID Register |

Appendix A AArch64 registers

This appendix contains the descriptions for the Neoverse™ V2 AArch64 registers.

A.1 AArch64 Generic system control register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** generic system control registers in the core. For more information about a register, you can click the register name in the table.

Table A-1: AArch64 Generic system control register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-----------------------------------|-----|-----|-----|-----|-----|---------------------------|--------|---|
| AIDR_EL1 | 3 | C0 | 1 | C0 | 7 | 0x0 | 64-bit | Auxiliary ID Register |
| ACTLR_EL1 | 3 | C1 | 0 | C0 | 1 | 0x0 | 64-bit | Auxiliary Control Register (EL1) |
| ACTLR_EL2 | 3 | C1 | 4 | C0 | 1 | 0x0 | 64-bit | Auxiliary Control Register (EL2) |
| HACR_EL2 | 3 | C1 | 4 | C1 | 7 | 0x0 | 64-bit | Hypervisor Auxiliary Control Register |
| ACTLR_EL3 | 3 | C1 | 6 | C0 | 1 | 0x0 | 64-bit | Auxiliary Control Register (EL3) |
| AMAIR_EL2 | 3 | C10 | 0 | C3 | 0 | 0x0 | 64-bit | Auxiliary Memory Attribute Indirection Register (EL2) |
| LORID_EL1 | 3 | C10 | 0 | C4 | 7 | See individual bit resets | 64-bit | LORegionID (EL1) |
| AMAIR_EL1 | 3 | C10 | 5 | C3 | 0 | 0x0 | 64-bit | Auxiliary Memory Attribute Indirection Register (EL1) |
| AMAIR_EL3 | 3 | C10 | 6 | C3 | 0 | 0x0 | 64-bit | Auxiliary Memory Attribute Indirection Register (EL3) |
| RMR_EL3 | 3 | C12 | 6 | C0 | 2 | See individual bit resets | 64-bit | Reset Management Register (EL3) |
| IMP_CPUACTLR_EL1 | 3 | C15 | 0 | C1 | 0 | See individual bit resets | 64-bit | CPU Auxiliary Control Register (EL1) |
| IMP_CPUACTLR2_EL1 | 3 | C15 | 0 | C1 | 1 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 2 (EL1) |
| IMP_CPUACTLR3_EL1 | 3 | C15 | 0 | C1 | 2 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 3 (EL1) |
| IMP_CPUACTLR4_EL1 | 3 | C15 | 0 | C1 | 3 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 4 (EL1) |
| IMP_CPUECTLR_EL1 | 3 | C15 | 0 | C1 | 4 | See individual bit resets | 64-bit | CPU Extended Control Register |
| IMP_CPUECTLR2_EL1 | 3 | C15 | 0 | C1 | 5 | See individual bit resets | 64-bit | CPU Extended Control Register 2 |
| IMP_CPUBUSQOS_EL1 | 3 | C15 | 0 | C1 | 7 | See individual bit resets | 64-bit | CPU Bus QoS Register |
| IMP_CPUPPMCR3_EL3 | 3 | C15 | 0 | C2 | 4 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-------------------------|-----|-----|-----|-----|-----|---------------------------|--------|---|
| IMP_CPUPMPDPCR_EL1 | 3 | C15 | 0 | C2 | 4 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUL2DIRTYLNCT_EL1 | 3 | C15 | 0 | C2 | 5 | 0x0 | 64-bit | CPU L2 Dirty Line Count Register |
| IMP_CPUPWRCTLR_EL1 | 3 | C15 | 0 | C2 | 7 | 0x0 | 64-bit | CPU Power Control Register |
| IMP_ATCR_EL1 | 3 | C15 | 0 | C7 | 0 | 0x0 | 64-bit | CPU Auxiliary Translation Control Register (EL1) |
| IMP_CPUACTLR5_EL1 | 3 | C15 | 0 | C8 | 0 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 5 (EL1) |
| IMP_CPUACTLR6_EL1 | 3 | C15 | 0 | C8 | 1 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 6 (EL1) |
| IMP_CPUACTLR7_EL1 | 3 | C15 | 0 | C8 | 2 | See individual bit resets | 64-bit | CPU Auxiliary Control Register 7 (EL1) |
| IMP_ATCR_EL2 | 3 | C15 | 4 | C7 | 0 | 0x0 | 64-bit | CPU Auxiliary Translation Control Register (EL2) |
| IMP_AVTCR_EL2 | 3 | C15 | 4 | C7 | 1 | 0x0 | 64-bit | CPU Virtualization Auxiliary Translation Control Register (EL2) |
| IMP_CPUPPMCR_EL3 | 3 | C15 | 6 | C2 | 0 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUMPMMCR_EL3 | 3 | C15 | 6 | C2 | 1 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUPPMCR2_EL3 | 3 | C15 | 6 | C2 | 1 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUL2SDIRTYLNCT_EL3 | 3 | C15 | 6 | C2 | 3 | 0x0 | 64-bit | CPU L2 Secure Dirty Line Count Register |
| IMP_CPUPDPTUNE_EL3 | 3 | C15 | 6 | C2 | 4 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUPPMCR4_EL3 | 3 | C15 | 6 | C2 | 4 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUPDPTUNE2_EL3 | 3 | C15 | 6 | C2 | 5 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUPPMCR5_EL3 | 3 | C15 | 6 | C2 | 5 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUMPMMTUNE_EL3 | 3 | C15 | 6 | C2 | 6 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUPPMCR6_EL3 | 3 | C15 | 6 | C2 | 6 | See individual bit resets | 64-bit | CPU Power Performance Management Control Register |
| IMP_CPUACTLR_EL3 | 3 | C15 | 6 | C4 | 0 | See individual bit resets | 64-bit | CPU Auxiliary Control Register (EL3) |
| IMP_ATCR_EL3 | 3 | C15 | 6 | C7 | 0 | 0x0 | 64-bit | CPU Auxiliary Translation Control Register (EL2) |
| IMP_CPUPSELR_EL3 | 3 | C15 | 6 | C8 | 0 | See individual bit resets | 64-bit | Selected Instruction Private Select Register |
| IMP_CPUPCR_EL3 | 3 | C15 | 6 | C8 | 1 | See individual bit resets | 64-bit | Selected Instruction Private Control Register |
| IMP_CPUPOR_EL3 | 3 | C15 | 6 | C8 | 2 | See individual bit resets | 64-bit | Selected Instruction Private Opcode Register |
| IMP_CPUPMR_EL3 | 3 | C15 | 6 | C8 | 3 | See individual bit resets | 64-bit | Selected Instruction Private Mask Register |
| IMP_CPUPOR2_EL3 | 3 | C15 | 6 | C8 | 4 | See individual bit resets | 64-bit | Selected Instruction Private Opcode Register 2 |

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-----------------|-----|-----|-----|-----|-----|---------------------------|--------|--|
| IMP_CPUPMR2_EL3 | 3 | C15 | 6 | C8 | 5 | See individual bit resets | 64-bit | Selected Instruction Private Mask Register 2 |
| IMP_CPUPFR_EL3 | 3 | C15 | 6 | C8 | 6 | See individual bit resets | 64-bit | Selected Instruction Private Flag Register |
| FPCR | 3 | C4 | 3 | C4 | 0 | See individual bit resets | 64-bit | Floating-point Control Register |
| AFSR0_EL2 | 3 | C5 | 0 | C1 | 0 | 0x0 | 64-bit | Auxiliary Fault Status Register 0 (EL2) |
| AFSR1_EL2 | 3 | C5 | 0 | C1 | 1 | 0x0 | 64-bit | Auxiliary Fault Status Register 1 (EL2) |
| AFSR0_EL1 | 3 | C5 | 5 | C1 | 0 | 0x0 | 64-bit | Auxiliary Fault Status Register 0 (EL1) |
| AFSR1_EL1 | 3 | C5 | 5 | C1 | 1 | 0x0 | 64-bit | Auxiliary Fault Status Register 1 (EL1) |
| AFSR0_EL3 | 3 | C5 | 6 | C1 | 0 | 0x0 | 64-bit | Auxiliary Fault Status Register 0 (EL3) |
| AFSR1_EL3 | 3 | C5 | 6 | C1 | 1 | 0x0 | 64-bit | Auxiliary Fault Status Register 1 (EL3) |

A.1.1 AIDR_EL1, Auxiliary ID Register

Provides **IMPLEMENTATION DEFINED** identification information.

The value of this register must be interpreted in conjunction with the value of AArch64-MIDR_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-1: AArch64_aidr_el1 bit assignments

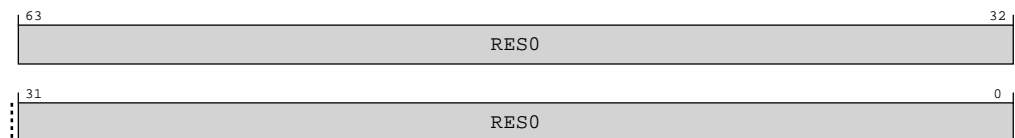


Table A-2: AIDR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, AIDR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AIDR_EL1 | 0b11 | 0b001 | 0b0000 | 0b0000 | 0b111 |

Accessibility

MRS <Xt>, AIDR_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AIDR_EL1;
elseif PSTATE.EL == EL2 then
    return AIDR_EL1;
elseif PSTATE.EL == EL3 then
    return AIDR_EL1;

```

A.1.2 ACTLR_EL1, Auxiliary Control Register (EL1)

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



Note

Arm recommends the contents of this register have no effect on the PE when AArch64-HCR_EL2.{E2H, TGE} is {1, 1}, and instead the configuration and control fields are provided by the AArch64-ACTLR_EL2 register. This avoids the need for software to manage the contents of these register when switching between a Guest OS and a Host OS.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-2: AArch64_actlr_el1 bit assignments

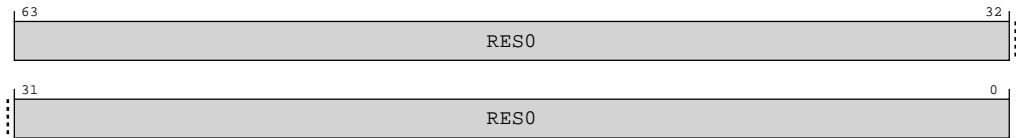


Table A-4: ACTLR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, ACTLR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ACTLR_EL1 | 0b11 | 0b000 | 0b0001 | 0b0000 | 0b001 |

MSR ACTLR_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ACTLR_EL1 | 0b11 | 0b000 | 0b0001 | 0b0000 | 0b001 |

Accessibility

MRS <Xt>, ACTLR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        return NVMem[0x118];
    else
        return ACTLR_EL1;
elseif PSTATE.EL == EL2 then
    return ACTLR_EL1;
elseif PSTATE.EL == EL3 then
    return ACTLR_EL1;

```

MSR ACTLR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        NVMem[0x118] = X[t];
    else
        ACTLR_EL1 = X[t];

```

```

elseif PSTATE.EL == EL2 then
    ACTLR_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ACTLR_EL1 = X[t];

```

A.1.3 ACTLR_EL2, Auxiliary Control Register (EL2)

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



Arm recommends the contents of this register are updated to apply to EL0 when AArch64-HCR_EL2.{E2H, TGE} is {1, 1}, gaining configuration and control fields from the AArch64-ACTLR_EL1. This avoids the need for software to manage the contents of these register when switching between a Guest OS and a Host OS.

Configurations

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

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-3: AArch64_actlr_el2 bit assignments

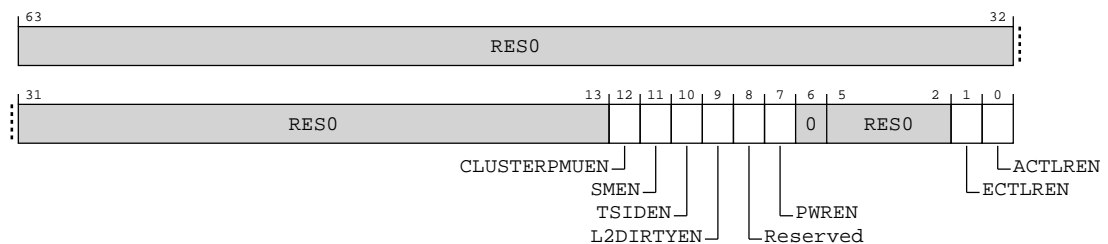


Table A-7: ACTLR_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|-------------|-------|
| [63:13] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|--------------|---|--------|
| [12] | CLUSTERPMUEN | Performance Management Registers enable. The possible values are: 0b0 CLUSTERPM* registers are not write-accessible from EL1. This is the reset value. 0b1 CLUSTERPM* registers are write-accessible from EL1 if they are write-accessible from EL2. | 0x0 |
| [11] | SMEN | Scheme Management Registers enable. The possible values are: 0b0 Registers CLUSTERACPSID, CLUSTERCFR2, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible EL1. This is the reset value. 0b1 Registers CLUSTERACPSID, CLUSTERCFR2, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible EL1 if they are write-accessible from EL2. | 0x0 |
| [10] | TSIDEN | Thread Scheme ID Register enable. The possible values are: 0b0 Register CLUSTERTHREADSID is not write-accessible from EL1. This is the reset value. 0b1 Register CLUSTERTHREADSID is write-accessible from EL1 if they are write-accessible from EL2 | 0x0 |
| [9] | L2DIRTYEN | L2 Dirty Line Count Register enable. The possible values are: 0b0 Register CPUL2DIRTYLNCT_EL1 is not read-accessible in EL1. This is the reset value. 0b1 Register CPUL2DIRTYLNCT_EL1 is read-accessible in EL1. | 0x0 |
| [8] | Reserved | Reserved 0b0 Reserved 0b1 Reserved | 0x0 |
| [7] | PWREN | Power Control Registers enable. The possible values are: 0b0 Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write accessible from EL1. This is the reset value. 0b1 Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible from EL1 if they are write-accessible from EL2 | 0x0 |
| [6:2] | RESO | Reserved | 0b0000 |

| Bits | Name | Description | Reset |
|------|--------|--|-------|
| [1] | ECTLRN | Extended Control Registers enable. The possible values are: 0b0 CPUECTLR*_EL1 and CLUSTERECTLR_EL1 are not write-accessible from EL1. This is the reset value. 0b1 CPUECTLR*_EL1 and CLUSTERECTLR_EL1 are write-accessible from EL1 if they are write-accessible from EL2. | 0b0 |
| [0] | ACTLRN | Auxiliary Control Registers enable. The possible values are: 0b0 CPUACTLR*_EL1 and CLUSTERACTLR are not write-accessible from EL1. This is the reset value. 0b1 CPUACTLR*_EL1 and CLUSTERACTLR are write-accessible from EL1 if they are write-accessible from EL2. | 0b0 |

Access

MRS <Xt>, ACTLR_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ACTLR_EL2 | 0b11 | 0b100 | 0b0001 | 0b0000 | 0b001 |

MSR ACTLR_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ACTLR_EL2 | 0b11 | 0b100 | 0b0001 | 0b0000 | 0b001 |

Accessibility

MRS <Xt>, ACTLR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    return ACTLR_EL2;
elseif PSTATE.EL == EL3 then
    return ACTLR_EL2;

```

MSR ACTLR_EL2, <Xt>

```


if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then

```

```
ACTLR_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    ACTLR_EL2 = X[t];
```

A.1.4 HACR_EL2, Hypervisor Auxiliary Control Register

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



Note

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

Configurations

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

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-4: AArch64_hacr_el2 bit assignments

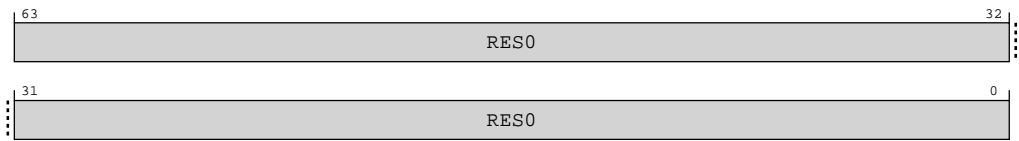


Table A-10: HACR_EL2 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, HACR_EL2

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

MSR HACR_EL2, <Xt>

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

Accessibility

MRS <Xt>, HACR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HACR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return HACR_EL2;
elsif PSTATE.EL == EL3 then
    return HACR_EL2;

```

MSR HACR_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HACR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HACR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    HACR_EL2 = X[t];

```

A.1.5 ACTLR_EL3, Auxiliary Control Register (EL3)

Provides **IMPLEMENTATION DEFINED** configuration and control options for EL3.

Configurations

This register is available in all configurations.

Attributes

Width

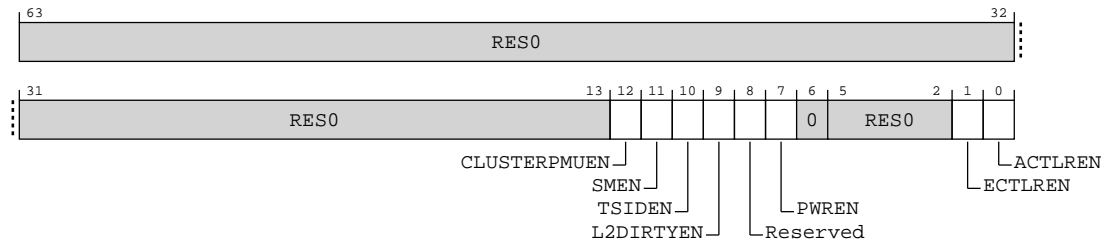
64

Functional group

Generic system control

Reset value

0x0

Bit descriptions**Figure A-5: AArch64_actlr_el3 bit assignments****Table A-13: ACTLR_EL3 bit descriptions**

| Bits | Name | Description | Reset |
|---------|--------------|---|-------|
| [63:13] | RES0 | Reserved | 0x0 |
| [12] | CLUSTERPMUEN | Performance Management Registers enable. The possible values are: 0b0 CLUSTERPM* registers are not write-accessible from EL2 and EL1. This is the reset value. 0b1 CLUSTERPM* registers are write-accessible from EL2 and EL1 if they are write-accessible from EL2. | 0x0 |
| [11] | SMEN | Scheme Management Registers enable. The possible values are: 0b0 Registers CLUSTERACPSID, CLUSTERCFR2, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible EL2 and EL1. This is the reset value. 0b1 Registers CLUSTERACPSID, CLUSTERCFR2, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible EL2 and EL1 if they are write-accessible from EL2. | 0x0 |
| [10] | TSIDEN | Thread Scheme ID Register enable. The possible values are: 0b0 Register CLUSTERTHREADSID is not write-accessible from EL2 and EL1. This is the reset value. 0b1 Register CLUSTERTHREADSID is write-accessible from EL2 and EL1 if they are write-accessible from EL2 | 0x0 |
| [9] | L2DIRTYEN | L2 Dirty Line Count Register enable. The possible values are: 0b0 Register CPUL2DIRTYLNCT_EL1 is not read-accessible in EL2 and EL1. This is the reset value. 0b1 Register CPUL2DIRTYLNCT_EL1 is read-accessible in EL2 and EL1. | 0x0 |

| Bits | Name | Description | Reset |
|-------|----------|--|--------|
| [8] | Reserved | Reserved 0b0 Reserved 0b1 Reserved | 0x0 |
| [7] | PWREN | Power Control Registers enable. The possible values are: 0b0 Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write accessible from EL2 and EL1. This is the reset value. 0b1 Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible from EL2 and EL1 if they are write-accessible from EL2 | 0x0 |
| [6:2] | RESO | Reserved | 0b0000 |
| [1] | ECTLREN | Extended Control Registers enable. The possible values are: 0b0 CPUECTLR*_EL2 and EL1 and CLUSTERECTLR_EL2 and EL1 are not write-accessible from EL2 and EL1. This is the reset value. 0b1 CPUECTLR*_EL2 and EL1 and CLUSTERECTLR_EL2 and EL1 are write-accessible from EL2 and EL1 if they are write-accessible from EL2. | 0b0 |
| [0] | ACTLREN | Auxiliary Control Registers enable. The possible values are: 0b0 CPUACTLR*_EL2 and EL1 and CLUSTERACTLR are not write-accessible from EL2 and EL1. This is the reset value. 0b1 CPUACTLR*_EL2 and EL1 and CLUSTERACTLR are write-accessible from EL2 and EL1 if they are write-accessible from EL2 | 0b0 |

Access

MRS <Xt>, ACTLR_EL3

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ACTLR_EL3 | 0b11 | 0b110 | 0b0001 | 0b0000 | 0b001 |

MSR ACTLR_EL3, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ACTLR_EL3 | 0b11 | 0b110 | 0b0001 | 0b0000 | 0b001 |

Accessibility

MRS <Xt>, ACTLR_EL3

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

```

    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return ACTLR_EL3;

```

MSR ACTLR_EL3, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    ACTLR_EL3 = X[t];

```

A.1.6 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by AArch64-MAIR_EL2.

Configurations

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

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

AMAIR_EL2 is permitted to be cached in a TLB.

Figure A-6: AArch64_amair_el2 bit assignments

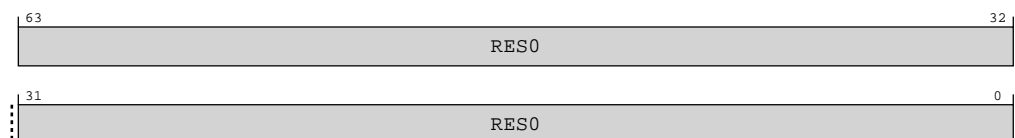


Table A-16: AMAIR_EL2 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, AMAIR_EL2

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

MSR AMAIR_EL2, <Xt>

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

MRS <Xt>, AMAIR_EL1

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

MSR AMAIR_EL1, <Xt>

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

Accessibility

MRS <Xt>, AMAIR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    return AMAIR_EL2;
elseif PSTATE.EL == EL3 then
    return AMAIR_EL2;

```

MSR AMAIR_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AMAIR_EL2 = X[t];
elseif PSTATE.EL == EL3 then

```

```
AMAIR_EL2 = X[t];
```

MRS <Xt>, AMAIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x148];
    else
        return AMAIR_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AMAIR_EL2;
    else
        return AMAIR_EL1;
elseif PSTATE.EL == EL3 then
    return AMAIR_EL1;
```

MSR AMAIR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x148] = X[t];
    else
        AMAIR_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL2 = X[t];
    else
        AMAIR_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    AMAIR_EL1 = X[t];
```

A.1.7 LORID_EL1, LORegionID (EL1)

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

Configurations

If no LORegion descriptors are implemented, then the registers AArch64-LORC_EL1, AArch64-LORN_EL1, AArch64-LOREA_EL1, and AArch64-LORSA_EL1 are RES0.

Attributes

Width

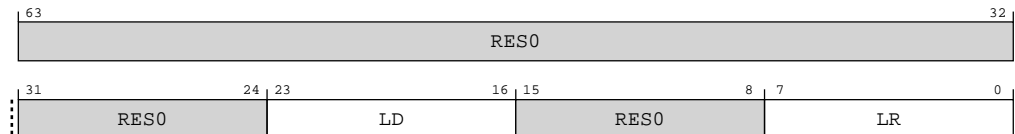
64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions**Figure A-7: AArch64_lorid_el1 bit assignments****Table A-21: LORID_EL1 bit descriptions**

| Bits | Name | Description | Reset |
|---------|------|---|------------|
| [63:24] | RES0 | Reserved | 0x0 |
| [23:16] | LD | Number of LORegion descriptors supported by the PE. This is an 8-bit binary number. 0b00000100 Four LOR descriptors are supported | |
| [15:8] | RES0 | Reserved | 0b00000000 |
| [7:0] | LR | Number of LORegions supported by the PE. This is an 8-bit binary number. Note: If LORID_EL1 indicates that no LORegions are implemented, then LoadLOAcquire and StoreLORelease will behave as LoadAcquire and StoreRelease. 0b00000100 Four LORegions are supported | |

Access

MRS <Xt>, LORID_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| LORID_EL1 | 0b11 | 0b000 | 0b1010 | 0b0100 | 0b111 |

Accessibility

MRS <Xt>, LORID_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.LORID_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TLOR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else

```

```
        return LORID_EL1;
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.TLOR == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return LORID_EL1;
    elsif PSTATE.EL == EL3 then
        return LORID_EL1;
```

A.1.8 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register (EL1)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by AArch64-MAIR_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

AMAIR_EL1 is permitted to be cached in a TLB.

Figure A-8: AArch64_amair_el1 bit assignments

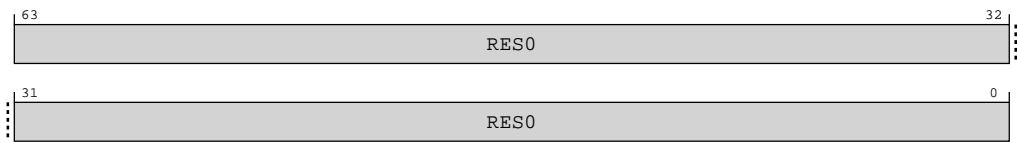


Table A-23: AMAIR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, AMAIR_EL1

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

MSR AMAIR_EL1, <Xt>

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

MRS <Xt>, AMAIR_EL12

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

MSR AMAIR_EL12, <Xt>

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

Accessibility

MRS <Xt>, AMAIR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x148];
    else
        return AMAIR_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AMAIR_EL2;
    else
        return AMAIR_EL1;
elseif PSTATE.EL == EL3 then
    return AMAIR_EL1;

```

MSR AMAIR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x148] = X[t];
    else
        AMAIR_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL2 = X[t];
    else
        AMAIR_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    AMAIR_EL1 = X[t];

```

MRS <Xt>, AMAIR_EL12

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        return NVMem[0x148];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AMAIR_EL1;
    else
        UNDEFINED;
elseif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        return AMAIR_EL1;
    else
        UNDEFINED;

```

MSR AMAIR_EL12, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x148] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t];
    else
        UNDEFINED;
elseif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t];
    else
        UNDEFINED;

```

A.1.9 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register (EL3)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by AArch64-MAIR_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

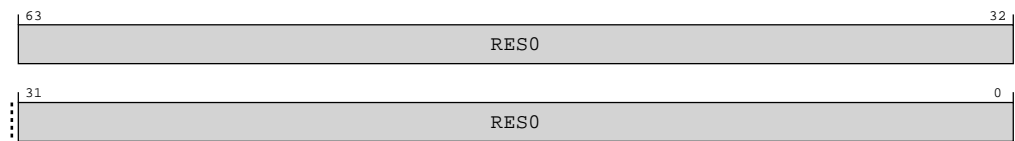
Generic system control

Reset value

0x0

Bit descriptions

AMAIR_EL3 is permitted to be cached in a TLB.

Figure A-9: AArch64_amair_el3 bit assignments**Table A-28: AMAIR_EL3 bit descriptions**

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, AMAIR_EL3

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AMAIR_EL3 | 0b11 | 0b110 | 0b1010 | 0b0011 | 0b000 |

MSR AMAIR_EL3, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AMAIR_EL3 | 0b11 | 0b110 | 0b1010 | 0b0011 | 0b000 |

Accessibility

MRS <Xt>, AMAIR_EL3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return AMAIR_EL3;

```

MSR AMAIR_EL3, <Xt>

```

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

```

```

    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    AMAIR_EL3 = X[t];

```

A.1.10 RMR_EL3, Reset Management Register (EL3)

A write to the register at EL3 can request a Warm reset.

Configurations

When EL3 is implemented:

- If EL3 can use all Execution states then this register must be implemented.
- In a AArch64 only implementation it is **IMPLEMENTATION DEFINED** whether the register is implemented.

Otherwise, direct accesses to RMR_EL3 are UNDEFINED.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-10: AArch64_rmr_el3 bit assignments

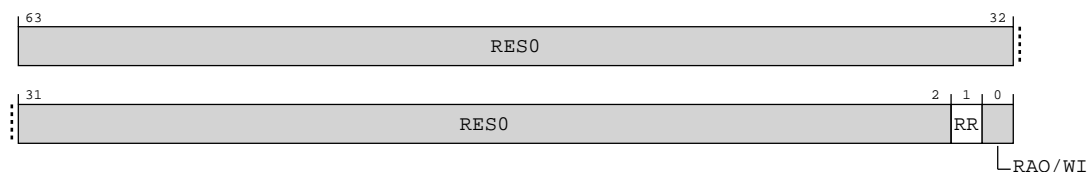


Table A-31: RMR_EL3 bit descriptions

| Bits | Name | Description | Reset |
|--------|--------|---|-------|
| [63:2] | RES0 | Reserved | 0x0 |
| [1] | RR | Reset Request. Setting this bit to 1 requests a Warm reset. | 0x0 |
| [0] | RAO/WI | Reserved | |

Access

MRS <Xt>, RMR_EL3

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| RMR_EL3 | 0b11 | 0b110 | 0b1100 | 0b0000 | 0b010 |

MSR RMR_EL3, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| RMR_EL3 | 0b11 | 0b110 | 0b1100 | 0b0000 | 0b010 |

Accessibility

MRS <Xt>, RMR_EL3

```

if PSTATE.EL == EL3 then
    return RMR_EL3;
else
    UNDEFINED;

```

MSR RMR_EL3, <Xt>

```

if PSTATE.EL == EL3 then
    RMR_EL3 = X[t];
else
    UNDEFINED;

```

A.1.11 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-11: AArch64_imp_cpuctlr_el1 bit assignments

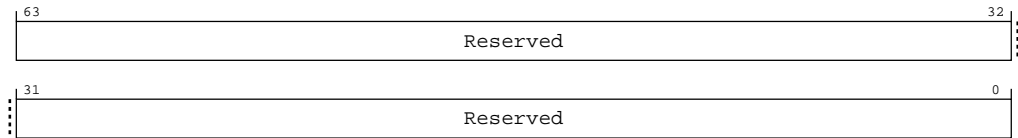


Table A-34: IMP_CPUACTLR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|----------|--------------------------------|-------|
| [63:0] | Reserved | Reserved for Arm internal use. | 0x0 |

Access

MRS <Xt>, S3_O_C15_C1_0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_O_C15_C1_0 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b000 |

MSR S3_O_C15_C1_0, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_O_C15_C1_0 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b000 |

Accessibility

MRS <Xt>, S3_O_C15_C1_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR_EL1;

```

MSR S3_O_C15_C1_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLRN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif ACTLR_EL3.ACTLRN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR_EL1 = X[t];

```

```
elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_CPUACTLR_EL1 = X[t];
```

A.1.12 IMP_CPUACTLR2_EL1, CPU Auxiliary Control Register 2 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-12: AArch64_imp_cpuactlr2_el1 bit assignments

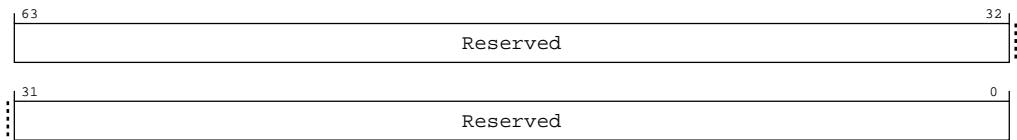


Table A-37: IMP_CPUACTLR2_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C1_1

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

MSR S3_0_C15_C1_1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_O_C15_C1_1 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b001 |

Accessibility

MRS <Xt>, S3_O_C15_C1_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR2_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR2_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR2_EL1;

```

MSR S3_O_C15_C1_1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLRN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif ACTLR_EL3.ACTLRN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLRN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_CPUACTLR2_EL1 = X[t];

```

A.1.13 IMP_CPUACTLR3_EL1, CPU Auxiliary Control Register 3 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

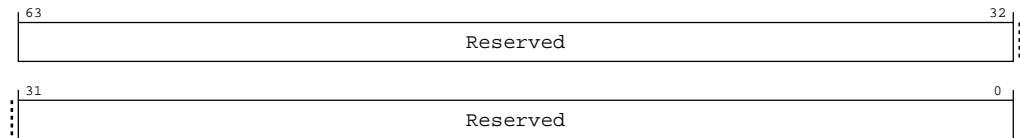
64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions**Figure A-13: AArch64_imp_cpuctlr3_el1 bit assignments****Table A-40: IMP_CPUACTLR3_EL1 bit descriptions**

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

Access

MRS <Xt>, S3_0_C15_C1_2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C1_2 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b010 |

MSR S3_0_C15_C1_2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C1_2 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b010 |

Accessibility

MRS <Xt>, S3_0_C15_C1_2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR3_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR3_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR3_EL1;

```

MSR S3_0_C15_C1_2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLRN == '0' then

```

```

    AArch64.SystemAccessTrap(EL2, 0x18);
elseif ACTLR_EL3.ACTLREN == '0' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    IMP_CPUACTLR3_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR3_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_CPUACTLR3_EL1 = X[t];

```

A.1.14 IMP_CPUACTLR4_EL1, CPU Auxiliary Control Register 4 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-14: AArch64_imp_cpuactlr4_el1 bit assignments

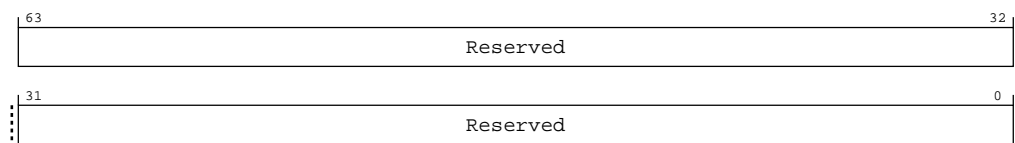


Table A-43: IMP_CPUACTLR4_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C1_3

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C1_3 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b011 |

MSR S3_0_C15_C1_3, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C1_3 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b011 |

Accessibility

MRS <Xt>, S3_0_C15_C1_3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR4_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR4_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR4_EL1;

```

MSR S3_0_C15_C1_3, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLRN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif ACTLR_EL3.ACTLRN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR4_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLRN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR4_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_CPUACTLR4_EL1 = X[t];

```

A.1.15 IMP_CPUACTLR4_EL1, CPU Extended Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-15: AArch64_imp_cpuctlr_el1 bit assignments

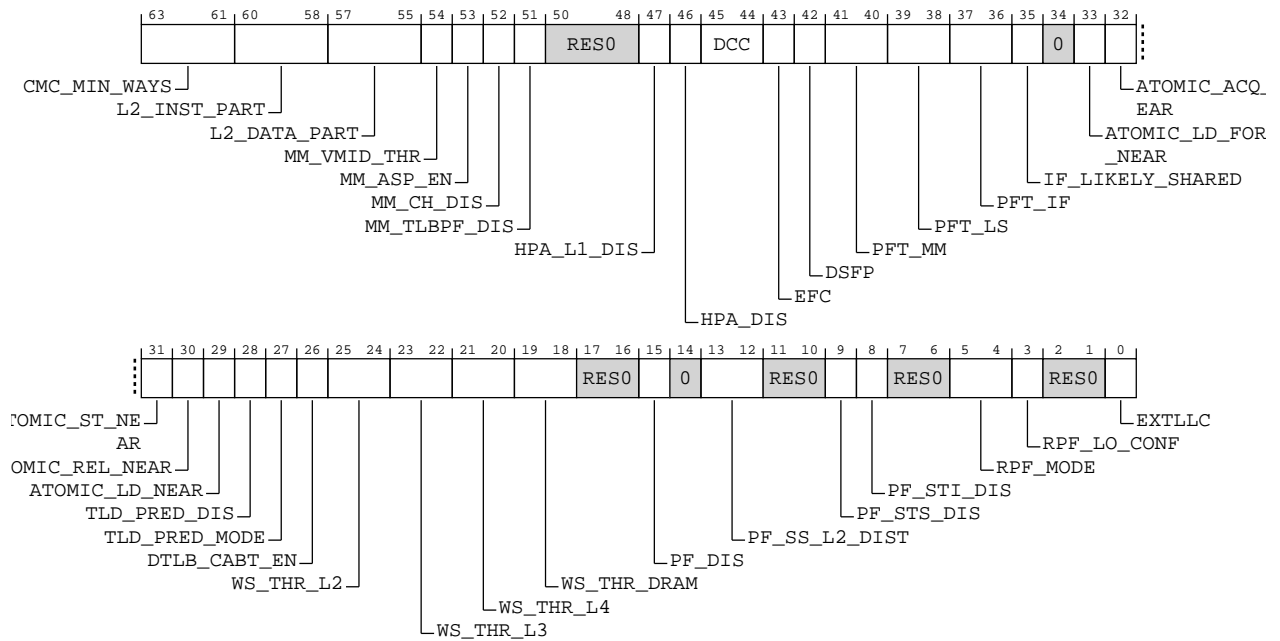


Table A-46: IMP_CPUCTL_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|--------------|--|-------|
| [63:61] | CMC_MIN_WAYS | Limits how many ways of L2 can be used by CMC. The possible values are: 0b000 CMC disabled 0b001 CMC must leave at least 1 way for data in L2 0b010 CMC must leave at least 2 ways for data in L2 - This is the default value. 0b011 CMC must leave at least 3 ways for data in L2 0b100 CMC must leave at least 4 ways for data in L2 0b101 CMC must leave at least 5 ways for data in L2 0b110 CMC must leave at least 6 ways for data in L2 0b111 CMC must leave at least 7 ways for data in L2 | |

| Bits | Name | Description | Reset |
|---------|--------------|--|-------|
| [60:58] | L2_INST_PART | <p>Partition the L2 cache for Instruction. The possible values are:</p> <p>0b000 No ways reserved for instructions. This is the reset value</p> <p>0b001 Reserve 1 way for instructions. Only instruction fetches can allocate way 7</p> <p>0b010 Reserve 2 ways for instructions. Only instruction fetches can allocate ways 7:6</p> <p>0b011 Reserve 3 ways for instructions. Only instruction fetches can allocate ways 7:5</p> <p>0b100 Reserve 4 ways for instructions. Only instruction fetches can allocate ways 7:4</p> <p>0b101 Reserve 5 ways for instructions. Only instruction fetches can allocate ways 7:3</p> <p>0b110 Reserve 6 ways for instructions. Only instruction fetches can allocate ways 7:2</p> <p>0b111 Reserve 7 ways for instructions. Only instruction fetches can allocate ways 7:1</p> | |
| [57:55] | L2_DATA_PART | <p>Reserve L2 capacity for data accesses. The possible values are:</p> <p>0b000 No ways reserved for data. This is the reset value</p> <p>0b001 Reserve 1 way for data. Only data accesses can allocate way 0</p> <p>0b010 Reserve 2 ways for data. Only data accesses can allocate ways 1:0</p> <p>0b011 Reserve 3 ways for data. Only data accesses can allocate ways 2:0</p> <p>0b100 Reserve 4 ways for data. Only data accesses can allocate ways 3:0</p> <p>0b101 Reserve 5 ways for data. Only data accesses can allocate ways 4:0</p> <p>0b110 Reserve 6 ways for data. Only data accesses can allocate ways 5:0</p> <p>0b111 Reserve 7 ways for data. Only data accesses can allocate ways 6:0</p> | |
| [54] | MM_VMID_THR | <p>VMID filter threshold. The possible values are:</p> <p>0b0 VMID filter flush after 16 unique VMID allocations to the MMU Translation Cache. This is the default value.</p> <p>0b1 VMID filter flush after 32 unique VMID allocations to the MMU Translation Cache</p> | |

| Bits | Name | Description | Reset |
|---------|--------------|--|-------|
| [53] | MM_ASP_EN | Disables allocation of splintered pages in L2 TLB. The possible values are: 0b0 Enables allocation of splintered pages in the L2 TLB. This is the default value. 0b1 Disables allocation of splintered pages in the L2 TLB. | |
| [52] | MM_CH_DIS | Disables use of contiguous hint. The possible values are: 0b0 Enables use of contiguous hint. This is the default value. 0b1 Disables use of contiguous hint. | |
| [51] | MM_TLBPF_DIS | Disables TLB prefetcher. The possible values are: 0b0 Enables TLB prefetcher. This is the default value. 0b1 Disables TLB prefetcher. | |
| [50:48] | RES0 | Reserved | 0b000 |
| [47] | HPA_L1_DIS | Disables hardware page aggregation in L1 TLBs. The possible values are: 0b0 Enables hardware page aggregation in L1 TLBs. This is the default value. 0b1 Disables hardware page aggregation in L1 TLBs. | |
| [46] | HPA_DIS | Disable Hardware page aggregation. The possible values are: 0b0 Enables hardware page aggregation. This is the default value. 0b1 Disables hardware page aggregation. | |
| [45:44] | DCC | Controls whether evictions of clean cache-lines send data on the CHI interface. Set this based on whether there is a cache on the path to memory. The possible values are: 0b00 Disables sending data when clean cache-lines are evicted. 0b01 Enables sending WriteEvictFull transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data. 0b10 Enables sending WriteEvictOrEvict transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data. This is the default value. 0b11 Enables sending WriteEvictOrEvict transactions when Unique Clean or Shared Clean cache-lines are evicted. | |

| Bits | Name | Description | Reset |
|---------|--------|--|-------|
| [43] | EFC | <p>Eviction Flush Control (EFC). Controls whether hardware cache flushes and DC CISCW instruction send data when evicting clean and dirty cache line. If it is known that data is likely to be used soon by another core, setting this bit can improve system performance. The possible values are:</p> <p>0b0</p> <p>Disables cache allocation in downstream caches when hardware cache flushes or DC CISCW instructions evict a clean or dirty cache line. Downstream Snoop Filter Present (DSFP) controls the sending of Evict transactions</p> <p>0b1</p> <p>Enables cache allocation in downstream caches when hardware cache flushes or DC CISCW instructions evict a clean or dirty cache line. Downstream Cache Control (DCC) controls the sending of data. DSFP controls the sending of Evict transactions.</p> | |
| [42] | DSFP | <p>Downstream Snoop Filter Present (DSFP). Enables sending Evict transactions on the CHI interface when clean lines are evicted without data. You must enable this if there is at least one snoop filter in the path to memory</p> <p>0b0</p> <p>Disables sending Evict transactions when clean cachelines are evicted without data</p> <p>0b1</p> <p>Enables sending of Evict transaction when clean cachelines are evicted without data.</p> | |
| [41:40] | PFT_MM | <p>DRAM prefetch using PrefetchTgt transactions for tablewalk requests. The possible values are:</p> <p>0b00</p> <p>Disable PrefetchTgt generation for requests from the Memory Management unit (MMU). This is the default value.</p> <p>0b01</p> <p>Conservatively generate PrefetchTgt for cacheable requests from the MMU, always generate for Non-cacheable.</p> <p>0b10</p> <p>Aggressively generate PrefetchTgt for cacheable requests from the MMU, always generate for Non-cacheable.</p> <p>0b11</p> <p>Always generate PrefetchTgt for cacheable requests from the MMU, always generate for Non-cacheable.</p> | |

| Bits | Name | Description | Reset |
|---------|----------------------|--|-------|
| [39:38] | PFT_LS | <p>DRAM prefetch using PrefetchTgt transactions for load and store requests. The possible values are:</p> <p>0b00 Disable PrefetchTgt generation for requests from the Load-Store unit (LS). This is the default value.</p> <p>0b01 Conservatively generate PrefetchTgt for cacheable requests from the LS, always generate for Non-cacheable.</p> <p>0b10 Aggressively generate PrefetchTgt for cacheable requests from the LS, always generate for Non-cacheable.</p> <p>0b11 Always generate PrefetchTgt for cacheable requests from the LS, always generate for Non-cacheable.</p> | |
| [37:36] | PFT_IF | <p>DRAM prefetch using PrefetchTgt transactions for instruction fetch requests. The possible values are:</p> <p>0b00 Disable PrefetchTgt generation for requests from the Instruction Fetch unit (IF). This is the default value.</p> <p>0b01 Conservatively generate PrefetchTgt for cacheable requests from the IF, always generate for Non-cacheable.</p> <p>0b10 Aggressively generate PrefetchTgt for cacheable requests from the IF, always generate for Non-cacheable.</p> <p>0b11 Always generate PrefetchTgt for cacheable requests from the IF, always generate for Non-cacheable.</p> | |
| [35] | IF_LIKELY_SHARED | <p>Instruction fetch Shared state control. The possible values are:</p> <p>0b0 Instruction fetch requests do not assert TXREQ LikelyShared. This is the reset value.</p> <p>0b1 Instruction fetch requests assert TXREQ LikelyShared and request a SharedClean copy of data.</p> | |
| [34] | RES0 | Reserved | 0b0 |
| [33] | ATOMIC_LD_FORCE_NEAR | <p>A load atomic (including SWP & CAS) instruction to WB memory will be performed near. The possible values are:</p> <p>0b0 Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request.</p> <p>0b1 Load-atomic will be performed near by bringing the line into the L1D Cache. This is the default value.</p> | |

| Bits | Name | Description | Reset |
|------|-----------------|--|-------|
| [32] | ATOMIC_ACQ_NEAR | <p>An atomic instruction to WB memory with acquire semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:</p> <p>0b0 Acquire-atomic is near if cache line is already Exclusive, otherwise make far atomic request.</p> <p>0b1 Acquire-atomic will make up to 1 fill request to perform near. This is the default value.</p> | |
| [31] | ATOMIC_ST_NEAR | <p>A store atomic instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:</p> <p>0b0 Store-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the default value.</p> <p>0b1 Store-atomic will make up to 1 fill request to perform near.</p> | |
| [30] | ATOMIC_REL_NEAR | <p>An atomic instruction to WB memory with release semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:</p> <p>0b0 Release-atomic is near if cache line is already Exclusive, otherwise make far atomic request.</p> <p>0b1 Release-atomic will make up to 1 fill request to perform near. This is the default value.</p> | |
| [29] | ATOMIC_LD_NEAR | <p>A load atomic (including SWP & CAS) instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:</p> <p>0b0 Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the default value.</p> <p>0b1 Load-atomic will make up to 1 fill request to perform near.</p> | |
| [28] | TLD_PRED_DIS | <p>Disable Transient Load Prediction. The possible values are:</p> <p>0b0 Enables transient load prediction. This is the default value.</p> <p>0b1 Disables transient load prediction.</p> | |
| [27] | TLD_PRED_MODE | <p>Aggressive Transient Load Prediction. The possible values are:</p> <p>0b0 Disables aggressive transient load prediction. This is the default value.</p> <p>0b1 Enables aggressive transient load prediction.</p> | |
| [26] | DTLB_CABT_EN | <p>Enables TLB Conflict Data Abort Exception. The possible values are:</p> <p>0b0 Disables TLB conflict data abort exception. This is the default value.</p> <p>0b1 Enables TLB conflict data abort exception.</p> | |

| Bits | Name | Description | Reset |
|---------|-------------|--|-------|
| [25:24] | WS_THR_L2 | <p>Threshold for direct stream to L2 cache on store. The possible values are:</p> <p>0b00 256B - This is the default value</p> <p>0b01 4KB</p> <p>0b10 8KB</p> <p>0b11 Disables direct stream to L2 cache on store.</p> | |
| [23:22] | WS_THR_L3 | <p>Threshold for direct stream to L3 cache on store. The possible values are:</p> <p>0b00 128KB</p> <p>0b01 256KB - This is the default value</p> <p>0b10 512KB</p> <p>0b11 Disables direct stream to L3 cache on store.</p> | |
| [21:20] | WS_THR_L4 | <p>Threshold for direct stream to L4 cache on store. The possible values are:</p> <p>0b00 256KB</p> <p>0b01 512KB - This is the default value</p> <p>0b10 1MB</p> <p>0b11 Disables direct stream to L4 cache on store.</p> | |
| [19:18] | WS_THR_DRAM | <p>Threshold for direct stream to DRAM on store. The possible values are:</p> <p>0b00 512KB</p> <p>0b01 1MB - This is the default value</p> <p>0b10 2MB</p> <p>0b11 Disables direct stream to DRAM on store.</p> | |
| [17:16] | RESO | Reserved | 0b00 |
| [15] | PF_DIS | <p>Disables hardware prefetching. The possible values are:</p> <p>0b0 Enables hardware prefetching. This is the default value.</p> <p>0b1 Disables hardware prefetching.</p> | |

| Bits | Name | Description | Reset |
|---------|---------------|---|-------|
| [14] | RES0 | Reserved | 0b0 |
| [13:12] | PF_SS_L2_DIST | Single cache line stride prefetching L2 distance. The possible values are: 0b00 22 lines ahead 0b01 40 lines ahead 0b10 60 lines ahead 0b11 Dynamic. This is the default value. | |
| [11:10] | RES0 | Reserved | 0b00 |
| [9] | PF_STS_DIS | Disable store-stride prefetches. The possible values are: 0b0 Enables store prefetching. This is the default value. 0b1 Disables store prefetching. | |
| [8] | PF_STI_DIS | Disable store prefetches at issue (not overridden by ls_hw_pref_disable). The possible values are: 0b0 Enables store prefetching. This is the default value. 0b1 Disable store prefetching. | |
| [7:6] | RES0 | Reserved | 0b00 |
| [5:4] | RPF_MODE | Region prefetcher aggressiveness. The possible values are: 0b00 Dynamic region prefetch aggressiveness. This is the default value. 0b01 Conservative region prefetching. 0b10 Very Conservative region prefetching. 0b11 Most Conservative region prefetching. This will disable the region prefetcher. | |
| [3] | RPF_LO_CONF | Region Prefetcher single accesses training behavior. The possible values are: 0b0 Mostly don't train PHT on single access. This is the default value. 0b1 Always train the PHT on single access. This results in fewer prefetch requests. | |
| [2:1] | RES0 | Reserved | 0b00 |

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [0] | EXTLLC | <p>Internal or external Last-level cache (LLC) in the system. The possible values are:</p> <p>0b0</p> <p>Indicates that an internal 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. This is the default value.</p> <p>0b1</p> <p>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.</p> | |

Access

MRS <Xt>, S3_O_C15_C1_4

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_O_C15_C1_4 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b100 |

MSR S3_O_C15_C1_4, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_O_C15_C1_4 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b100 |

Accessibility

MRS <Xt>, S3_O_C15_C1_4

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUECTLR_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUECTLR_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUECTLR_EL1;

```

MSR S3_O_C15_C1_4, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else

```

```
IMP_CPUECTLR_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_CPUECTLR_EL1 = X[t];
```

A.1.16 IMP_CPUECTLR2_EL1, CPU Extended Control Register 2

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-16: AArch64_imp_cpuectlr2_el1 bit assignments

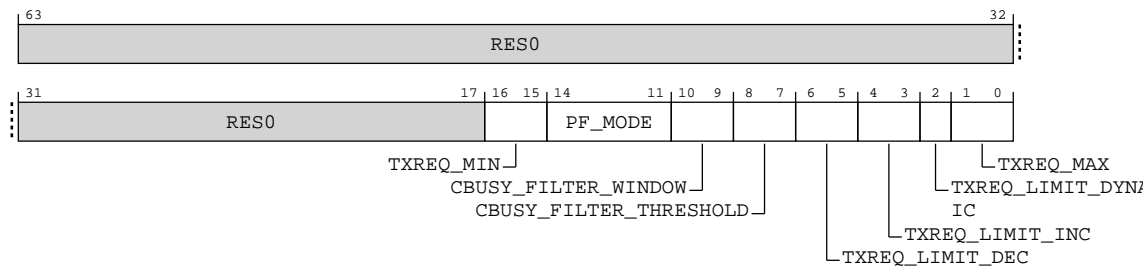


Table A-49: IMP_CPUECTLR2_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|-------------|-------|
| [63:17] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|---------|-----------|--|-------|
| [16:15] | TXREQ_MIN | <p>Minimum number of TXREQ transactions outstanding from the L2 Transaction Queue. The possible values are:</p> <p>0b00 1/4 of L2 TQ size - This is the default value.</p> <p>0b01 1/8 of L2 TQ size</p> <p>0b10 1/16 of L2 TQ size</p> <p>0b11 1/32 of L2 TQ size</p> | |

| Bits | Name | Description | Reset |
|---------|---------|--|-------|
| [14:11] | PF_MODE | <p>Prefetcher Aggressiveness Modes. With mode 0 representing the most aggressive mode and 3 representing the most conservative mode. The possible values and associated ranges are:</p> <p>0b0000 Modes [0,0] (statically at the most aggressive mode)</p> <p>0b0001 Modes [0,1]</p> <p>0b0010 Modes [0,2]</p> <p>0b0011 Modes [0,3] - This is the default value.</p> <p>0b0100 Modes [1,1]</p> <p>0b0101 Modes [1,2]</p> <p>0b0110 Modes [1,3]</p> <p>0b0111 Modes [2,2]</p> <p>0b1000 Modes [2,3]</p> <p>0b1001 Modes [3,3] (statically at the most conservative mode)</p> <p>0b1010 reserved</p> <p>0b1011 reserved</p> <p>0b1100 reserved</p> <p>0b1101 reserved</p> <p>0b1110 reserved</p> <p>0b1111 reserved</p> | |

| Bits | Name | Description | Reset |
|--------|------------------------|--|-------|
| [10:9] | CBUSY_FILTER_WINDOW | <p>Number of CBusy responses in one sampling window. The possible values are:</p> <p>0b00 256 - This is the default value</p> <p>0b01 64</p> <p>0b10 128</p> <p>0b11 512</p> | |
| [8:7] | CBUSY_FILTER_THRESHOLD | <p>Fraction of of CBusy responses in the sampling window necessary to be considered a valid sample of that CBusy value. The possible values are:</p> <p>0b00 1/16 - This is the default value</p> <p>0b01 1/32</p> <p>0b10 1/8</p> <p>0b11 1/4</p> | |
| [6:5] | TXREQ_LIMIT_DEC | <p>Dynamic TXREQ limit decrement. Controls how quickly the dynamic TXREQ limit is decreased when CBusy indicates value of 3. The possible values are:</p> <p>0b00 4 - This is the default value</p> <p>0b01 8</p> <p>0b10 16</p> <p>0b11 2</p> | |
| [4:3] | TXREQ_LIMIT_INC | <p>Dynamic TXREQ limit increment. Controls how quickly the dynamic TXREQ limit is increased when CBusy indicates values less than 2. The possible values are:</p> <p>0b00 4 - This is the default value</p> <p>0b01 8</p> <p>0b10 16</p> <p>0b11 2</p> | |

| Bits | Name | Description | Reset |
|-------|---------------------|---|-------|
| [2] | TXREQ_LIMIT_DYNAMIC | <p>Selects static or dynamic control of TXREQ limit. Dynamic TXREQ limit will adjust based on CBusy responses on RXDAT and RXRSP in the range of the static limit selected by CPUECTLR2_EL1[1:0] and 1/4 of the L2 TQ SIZE. The possible values are:</p> <p>0b0 maximum number of TXREQ transactions statically set by CPUECTLR2_EL1[1:0] - This is the default value.</p> <p>0b1 maximum number of TXREQ transactions dynamically controlled</p> | |
| [1:0] | TXREQ_MAX | <p>Maximum number of TXREQ transactions outstanding from the L2 Transaction Queue. The possible values are:</p> <p>0b00 full L2 TQ size - This is the default value</p> <p>0b01 3/4 of L2 TQ size</p> <p>0b10 1/2 of L2 TQ size</p> <p>0b11 1/4 of L2 TQ size</p> | |

Access

MRS <Xt>, S3_0_C15_C1_5

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C1_5 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b101 |

MSR S3_0_C15_C1_5, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C1_5 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b101 |

Accessibility

MRS <Xt>, S3_0_C15_C1_5

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUECTLR2_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUECTLR2_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUECTLR2_EL1;

```

MSR S3_0_C15_C1_5, <Xt>

```

if PSTATE.EL == EL0 then

```

```

    UNDEFINED;
  elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ECTLREN == '0' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ECTLREN == '0' then
      AArch64.SystemAccessTrap(EL3, 0x18);
    else
      IMP_CPUECTLR2_EL1 = X[t];
    elsif PSTATE.EL == EL2 then
      if ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
      else
        IMP_CPUECTLR2_EL1 = X[t];
    elsif PSTATE.EL == EL3 then
      IMP_CPUECTLR2_EL1 = X[t];

```

A.1.17 IMP_CPUBUSQOS_EL1, CPU Bus QoS Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-17: AArch64_imp_cpibusqos_el1 bit assignments

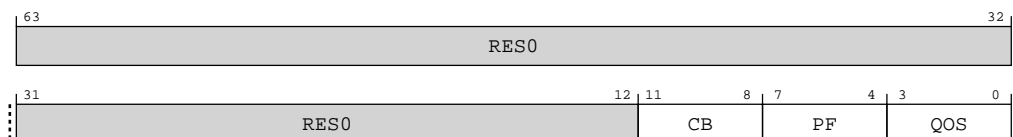


Table A-52: IMP_CPUBUSQOS_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|--|-------|
| [63:12] | RES0 | Reserved | 0x0 |
| [11:8] | CB | TXREQ QoS value used for copybacks. 0b1110 | |

| Bits | Name | Description | Reset |
|-------|------|---|-------|
| [7:4] | PF | Value driven on TXREQ QoS for prefetch accesses. 0b1010 | |
| [3:0] | QOS | Value driven on TXREQ QoS field for demand accesses. 0b1110 | |

Access

MRS <Xt>, S3_O_C15_C1_7

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_O_C15_C1_7 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b111 |

MSR S3_O_C15_C1_7, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_O_C15_C1_7 | 0b11 | 0b000 | 0b1111 | 0b0001 | 0b111 |

Accessibility

MRS <Xt>, S3_O_C15_C1_7

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUBUSQOS_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUBUSQOS_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUBUSQOS_EL1;

```

MSR S3_O_C15_C1_7, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.SMEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif ACTLR_EL3.SMEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUBUSQOS_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.SMEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUBUSQOS_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_CPUBUSQOS_EL1 = X[t];

```

A.1.18 IMP_CPUPPMCR3_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-18: AArch64_imp_cpuppmcr3_el3 bit assignments

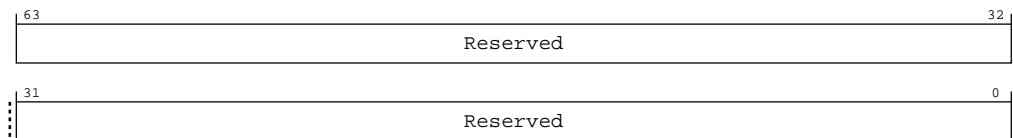


Table A-55: IMP_CPUPPMCR3_EL3 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C2_4

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C2_4 | 0b11 | 0b000 | 0b1111 | 0b0010 | 0b100 |

MSR S3_0_C15_C2_4, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C2_4 | 0b11 | 0b000 | 0b1111 | 0b0010 | 0b100 |

Accessibility

MRS <Xt>, S3_0_C15_C2_4

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

```
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR3_EL3;
```

MSR S3_0_C15_C2_4, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPPMCR3_EL3 = X[t];
```

A.1.19 IMP_CPUPPMPCR_EL1, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-19: AArch64_imp_cpuppmdpcr_el1 bit assignments

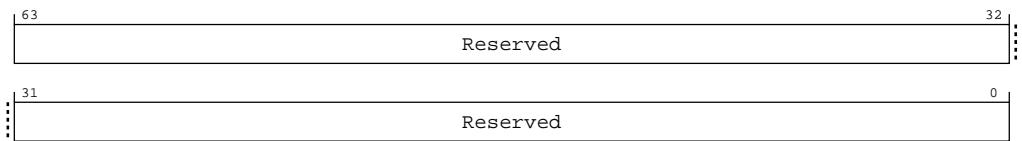


Table A-58: IMP_CPUPPMPCR_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C2_4

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C2_4 | 0b11 | 0b000 | 0b1111 | 0b0010 | 0b100 |

MSR S3_0_C15_C2_4, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C2_4 | 0b11 | 0b000 | 0b1111 | 0b0010 | 0b100 |

Accessibility

MRS <Xt>, S3_0_C15_C2_4

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPPMPDPCR_EL1;

```

MSR S3_0_C15_C2_4, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPPMPDPCR_EL1 = X[t];

```

A.1.20 IMP_CPUL2DIRTYLNCT_EL1, CPU L2 Dirty Line Count Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-20: AArch64_imp_cpul2dirtylnct_el1 bit assignments

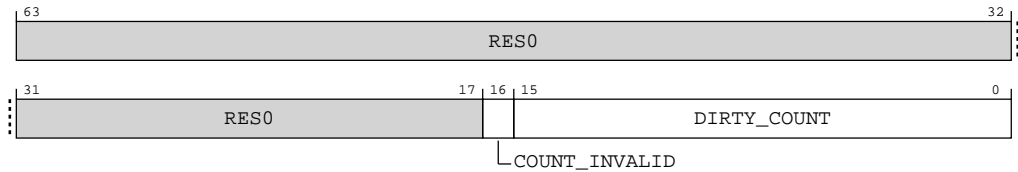


Table A-61: IMP_CPUL2DIRTYLNCT_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|---------------|--|-------|
| [63:17] | RES0 | Reserved | 0x0 |
| [16] | COUNT_INVALID | Indicates the dirty count is invalid. Reset value is 'b0 | 0x0 |
| [15:0] | DIRTY_COUNT | Number of dirty lines in the L2. Reset value is 'h0000 | 0x0 |

Access

MRS <Xt>, S3_O_C15_C2_5

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_O_C15_C2_5 | 0b11 | 0b000 | 0b1111 | 0b0010 | 0b101 |

Accessibility

MRS <Xt>, S3_O_C15_C2_5

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUL2DIRTYLNCT_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUL2DIRTYLNCT_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUL2DIRTYLNCT_EL1;

```

A.1.21 IMP_CPUPWRCTLR_EL1, CPU Power Control Register

This register controls various power aspects of the core.

Configurations

This register is available in all configurations.

Attributes

Width

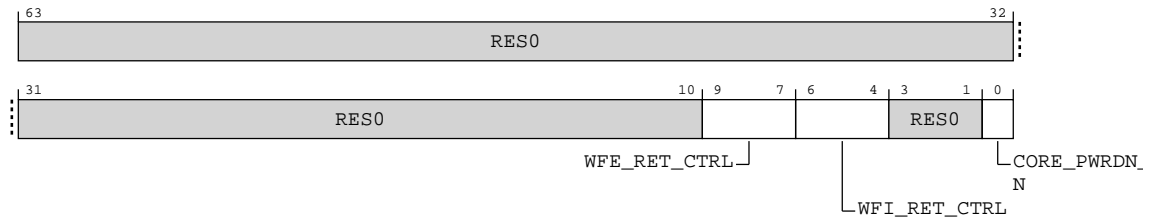
64

Functional group

Generic system control

Reset value

0x0

Bit descriptions**Figure A-21: AArch64_imp_cpupwrctrl_el1 bit assignments****Table A-63: IMP_CPUPWRCTLR_EL1 bit descriptions**

| Bits | Name | Description | Reset |
|---------|--------------|---|-------|
| [63:10] | RES0 | Reserved | 0x0 |
| [9:7] | WFE_RET_CTRL | Wait for Event retention control. The possible values are: 0b000 Dynamic retention is disabled. 0b001 2 system counter ticks are required before retention entry. 0b010 8 system counter ticks are required before retention entry. 0b011 32 system counter ticks are required before retention entry. 0b100 64 system counter ticks are required before retention entry. 0b101 128 system counter ticks are required before retention entry. 0b110 256 system counter ticks are required before retention entry. 0b111 512 system counter ticks are required before retention entry. | 0x0 |

| Bits | Name | Description | Reset |
|-------|---------------|---|-------|
| [6:4] | WFI_RET_CTRL | Wait for Interrupt retention control. The possible values are: 0b000 Dynamic retention is disabled. 0b001 2 system counter ticks are required before retention entry. 0b010 8 system counter ticks are required before retention entry. 0b011 32 system counter ticks are required before retention entry. 0b100 64 system counter ticks are required before retention entry. 0b101 128 system counter ticks are required before retention entry. 0b110 256 system counter ticks are required before retention entry. 0b111 512 system counter ticks are required before retention entry. | 0x0 |
| [3:1] | RESO | Reserved | 0b000 |
| [0] | CORE_PWRDN_EN | Indicates to the power controller if the CPU wants to power down when it enters WFE/WFI state. The possible values are: 0b0 CPU does not want to power down when it enters WFE/WFI state. 0b1 CPU wants to power down when it enters WFE/WFI state. | 0b0 |

Access

MRS <Xt>, S3_0_C15_C2_7

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C2_7 | 0b11 | 0b000 | 0b1111 | 0b0010 | 0b111 |

MSR S3_0_C15_C2_7, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C2_7 | 0b11 | 0b000 | 0b1111 | 0b0010 | 0b111 |

Accessibility

MRS <Xt>, S3_0_C15_C2_7

```

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

```

```

        return IMP_CPUPWRCTLR_EL1;
    elsif PSTATE.EL == EL2 then
        return IMP_CPUPWRCTLR_EL1;
    elsif PSTATE.EL == EL3 then
        return IMP_CPUPWRCTLR_EL1;

```

MSR S3_0_C15_C2_7, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.PWREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.PWREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPWRCTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.PWREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPWRCTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUPWRCTLR_EL1 = X[t];

```

A.1.22 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register (EL1)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-22: AArch64_imp_atcr_el1 bit assignments

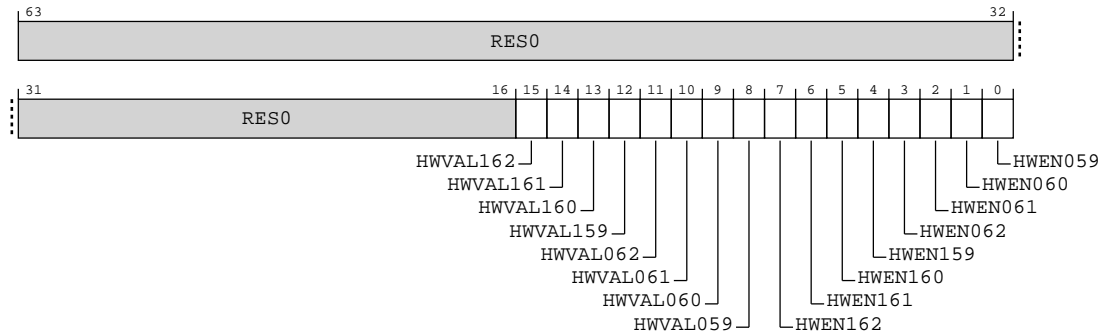


Table A-66: IMP_ATCR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|--|-------|
| [63:16] | RES0 | Reserved | 0x0 |
| [15] | HWVAL162 | Value of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN162 is set. | 0x0 |
| [14] | HWVAL161 | Value of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN161 is set. | 0x0 |
| [13] | HWVAL160 | Value of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN160 is set. | 0x0 |
| [12] | HWVAL159 | Value of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN159 is set. | 0x0 |
| [11] | HWVAL062 | Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL1 if HWEN062 is set. | 0x0 |
| [10] | HWVAL061 | Value of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL1 if HWEN061 is set. | 0x0 |
| [9] | HWVAL060 | Value of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL1 if HWEN060 is set. | 0x0 |
| [8] | HWVAL059 | Value of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL1 if HWEN059 is set. | 0x0 |
| [7] | HWEN162 | Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[3] will be 0 on translation table walks. | 0x0 |
| [6] | HWEN161 | Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[2] will be 0 on translation table walks. | 0x0 |
| [5] | HWEN160 | Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[1] will be 0 on translation table walks. | 0x0 |
| [4] | HWEN159 | Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[0] will be 0 on translation table walks. | 0x0 |
| [3] | HWEN062 | Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL1. If this bit is clear, PBHA[3] will be 0 on translation table walks. | 0x0 |
| [2] | HWEN061 | Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL1. If this bit is clear, PBHA[2] will be 0 on translation table walks. | 0x0 |
| [1] | HWEN060 | Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL1. If this bit is clear, PBHA[1] will be 0 on translation table walks. | 0x0 |
| [0] | HWEN059 | Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL1. If this bit is clear, PBHA[0] will be 0 on translation table walks. | 0x0 |

Access

MRS <Xt>, S3_0_C15_C7_0

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

MSR S3_0_C15_C7_0, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C7_0

```

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

```

MSR S3_0_C15_C7_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_ATCR_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    IMP_ATCR_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_ATCR_EL1 = X[t];

```

A.1.23 IMP_CPUACTLR5_EL1, CPU Auxiliary Control Register 5 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

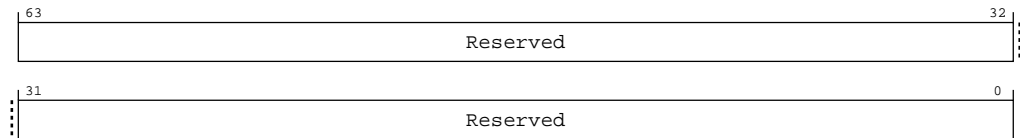
64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions**Figure A-23: AArch64_imp_cpuctlr5_el1 bit assignments****Table A-69: IMP_CPUACTLR5_EL1 bit descriptions**

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

Access

MRS <Xt>, S3_0_C15_C8_0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C8_0 | 0b11 | 0b000 | 0b1111 | 0b1000 | 0b000 |

MSR S3_0_C15_C8_0, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C8_0 | 0b11 | 0b000 | 0b1111 | 0b1000 | 0b000 |

Accessibility

MRS <Xt>, S3_0_C15_C8_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR5_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR5_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR5_EL1;

```

MSR S3_0_C15_C8_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLRN == '0' then

```

```

    AArch64.SystemAccessTrap(EL2, 0x18);
elseif ACTLR_EL3.ACTLREN == '0' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    IMP_CPUACTLR5_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR5_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_CPUACTLR5_EL1 = X[t];

```

A.1.24 IMP_CPUACTLR6_EL1, CPU Auxiliary Control Register 6 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-24: AArch64_imp_cpuactlr6_el1 bit assignments

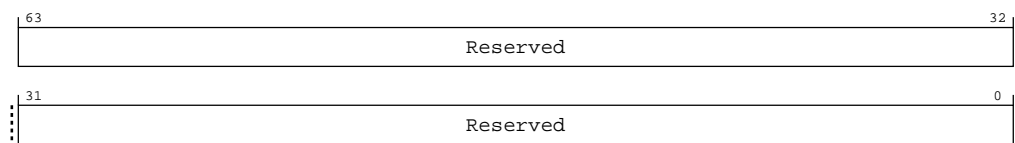


Table A-72: IMP_CPUACTLR6_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C8_1 | 0b11 | 0b000 | 0b1111 | 0b1000 | 0b001 |

MSR S3_0_C15_C8_1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C8_1 | 0b11 | 0b000 | 0b1111 | 0b1000 | 0b001 |

Accessibility

MRS <Xt>, S3_0_C15_C8_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR6_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR6_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR6_EL1;

```

MSR S3_0_C15_C8_1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR6_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR6_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_CPUACTLR6_EL1 = X[t];

```

A.1.25 IMP_CPUACTLR7_EL1, CPU Auxiliary Control Register 7 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

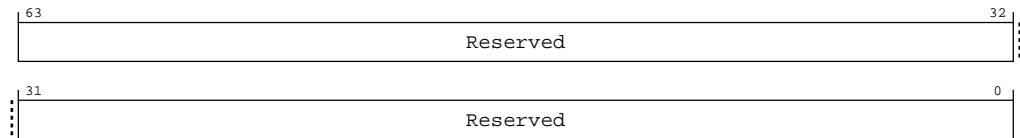
64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions**Figure A-25: AArch64_imp_cpuctlr7_el1 bit assignments****Table A-75: IMP_CPUACTLR7_EL1 bit descriptions**

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

Access

MRS <Xt>, S3_0_C15_C8_2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C8_2 | 0b11 | 0b000 | 0b1111 | 0b1000 | 0b010 |

MSR S3_0_C15_C8_2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C8_2 | 0b11 | 0b000 | 0b1111 | 0b1000 | 0b010 |

Accessibility

MRS <Xt>, S3_0_C15_C8_2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR7_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR7_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR7_EL1;

```

MSR S3_0_C15_C8_2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLRN == '0' then

```

```

    AArch64.SystemAccessTrap(EL2, 0x18);
elseif ACTLR_EL3.ACTLREN == '0' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    IMP_CPUACTLR7_EL1 = X[t];
elseif PSTATE_EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR7_EL1 = X[t];
elseif PSTATE_EL == EL3 then
    IMP_CPUACTLR7_EL1 = X[t];

```

A.1.26 IMP_ATCR_EL2, CPU Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-26: AArch64_imp_atcr_el2 bit assignments

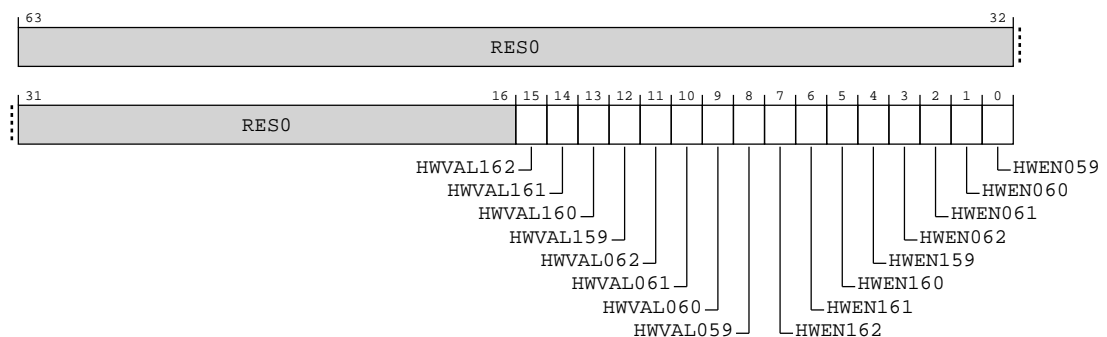


Table A-78: IMP_ATCR_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [63:16] | RES0 | Reserved | 0x0 |
| [15] | HWVAL162 | Value of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN162 is set. | 0x0 |
| [14] | HWVAL161 | Value of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN161 is set. | 0x0 |

| Bits | Name | Description | Reset |
|------|----------|--|-------|
| [13] | HWVAL160 | Value of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN160 is set. | 0x0 |
| [12] | HWVAL159 | Value of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN159 is set. | 0x0 |
| [11] | HWVAL062 | Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN062 is set. | 0x0 |
| [10] | HWVAL061 | Value of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN061 is set. | 0x0 |
| [9] | HWVAL060 | Value of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN060 is set. | 0x0 |
| [8] | HWVAL059 | Value of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN059 is set. | 0x0 |
| [7] | HWEN162 | Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks. | 0x0 |
| [6] | HWEN161 | Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks. | 0x0 |
| [5] | HWEN160 | Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks. | 0x0 |
| [4] | HWEN159 | Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks. | 0x0 |
| [3] | HWEN062 | Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks. | 0x0 |
| [2] | HWEN061 | Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks. | 0x0 |
| [1] | HWEN060 | Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks. | 0x0 |
| [0] | HWEN059 | Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks. | 0x0 |

Access

MRS <Xt>, S3_4_C15_C7_0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_4_C15_C7_0 | 0b11 | 0b100 | 0b1111 | 0b0111 | 0b000 |

MSR S3_4_C15_C7_0, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_4_C15_C7_0 | 0b11 | 0b100 | 0b1111 | 0b0111 | 0b000 |

Accessibility

MRS <Xt>, S3_4_C15_C7_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    return IMP_ATCR_EL2;
elseif PSTATE.EL == EL3 then
    return IMP_ATCR_EL2;

```

MSR S3_4_C15_C7_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_ATCR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL2 = X[t];

```

A.1.27 IMP_AVTCR_EL2, CPU Virtualization Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-27: AArch64_imp_avtcr_el2 bit assignments

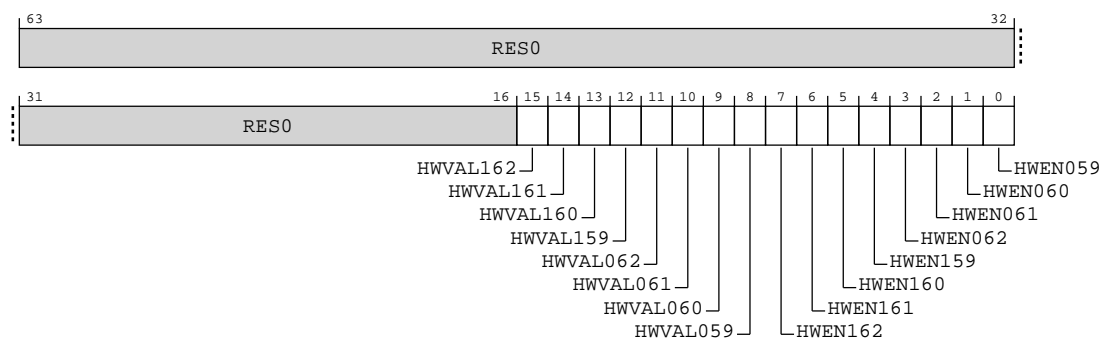


Table A-81: IMP_AVTCR_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [63:16] | RES0 | Reserved | 0x0 |
| [15] | HWVAL162 | Value of PBHA[3] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN162 is set. | 0x0 |
| [14] | HWVAL161 | Value of PBHA[2] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN161 is set. | 0x0 |
| [13] | HWVAL160 | Value of PBHA[1] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN160 is set. | 0x0 |
| [12] | HWVAL159 | Value of PBHA[0] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN159 is set. | 0x0 |
| [11] | HWVAL062 | Value of PBHA[3] on memory accesses due to translation table walks using TTBRO_EL2 if HWEN062 is set. | 0x0 |
| [10] | HWVAL061 | Value of PBHA[2] on memory accesses due to translation table walks using TTBRO_EL2 if HWEN061 is set. | 0x0 |
| [9] | HWVAL060 | Value of PBHA[1] on memory accesses due to translation table walks using TTBRO_EL2 if HWEN060 is set. | 0x0 |
| [8] | HWVAL059 | Value of PBHA[0] on memory accesses due to translation table walks using TTBRO_EL2 if HWEN059 is set. | 0x0 |
| [7] | HWEN162 | Enable use of PBHA[3] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks. | 0x0 |
| [6] | HWEN161 | Enable use of PBHA[2] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks. | 0x0 |
| [5] | HWEN160 | Enable use of PBHA[1] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks. | 0x0 |
| [4] | HWEN159 | Enable use of PBHA[0] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks. | 0x0 |
| [3] | HWEN062 | Enable use of PBHA[3] on memory accesses due to translation table walks using TTBRO_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks. | 0x0 |
| [2] | HWEN061 | Enable use of PBHA[2] on memory accesses due to translation table walks using TTBRO_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks. | 0x0 |
| [1] | HWEN060 | Enable use of PBHA[1] on memory accesses due to translation table walks using TTBRO_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks. | 0x0 |
| [0] | HWEN059 | Enable use of PBHA[0] on memory accesses due to translation table walks using TTBRO_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks. | 0x0 |

Access

MRS <Xt>, S3_4_C15_C7_1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_4_C15_C7_1 | 0b11 | 0b100 | 0b1111 | 0b0111 | 0b001 |

MSR S3_4_C15_C7_1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_4_C15_C7_1 | 0b11 | 0b100 | 0b1111 | 0b0111 | 0b001 |

Accessibility

MRS <Xt>, S3_4_C15_C7_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return IMP_AVTCR_EL2;
elsif PSTATE.EL == EL3 then
    return IMP_AVTCR_EL2;

```

MSR S3_4_C15_C7_1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_AVTCR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_AVTCR_EL2 = X[t];

```

A.1.28 IMP_CPUPPMCR_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-28: AArch64_imp_cpuppmcr_el3 bit assignments

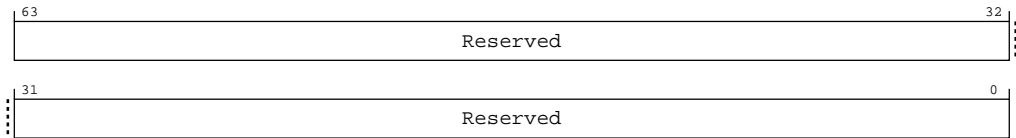


Table A-84: IMP_CPUPPMCR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_0 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b000 |

MSR S3_6_C15_C2_0, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_0 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b000 |

Accessibility

MRS <Xt>, S3_6_C15_C2_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR_EL3;

```

MSR S3_6_C15_C2_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPPMCR_EL3 = X[t];

```

A.1.29 IMP_CPUMPMMCR_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-29: AArch64_imp_cpumpmmcr_el3 bit assignments

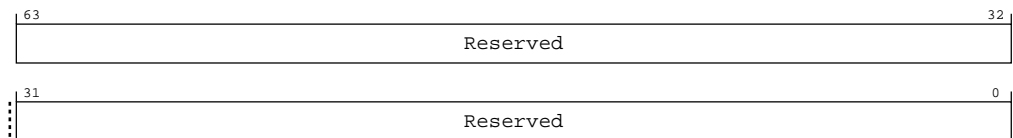


Table A-87: IMP_CPUMPMMCR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_1 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b001 |

MSR S3_6_C15_C2_1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_1 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b001 |

Accessibility

MRS <Xt>, S3_6_C15_C2_1

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

```
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR_EL3;
```

MSR S3_6_C15_C2_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPPMCR_EL3 = X[t];
```

A.1.30 IMP_CPUPPMCR2_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-30: AArch64_imp_cpuppmcr2_el3 bit assignments

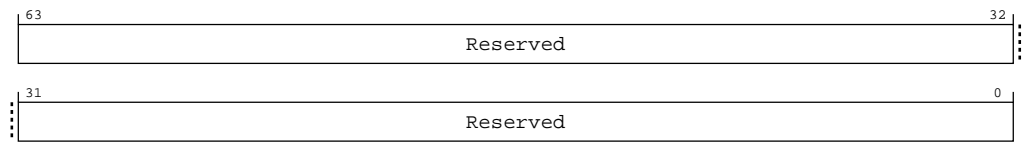


Table A-90: IMP_CPUPPMCR2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_1 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b001 |

MSR S3_6_C15_C2_1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_1 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b001 |

Accessibility

MRS <Xt>, S3_6_C15_C2_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR2_EL3;

```

MSR S3_6_C15_C2_1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPPMCR2_EL3 = X[t];

```

A.1.31 IMP_CPUL2SDIRTYLNCT_EL3, CPU L2 Secure Dirty Line Count Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

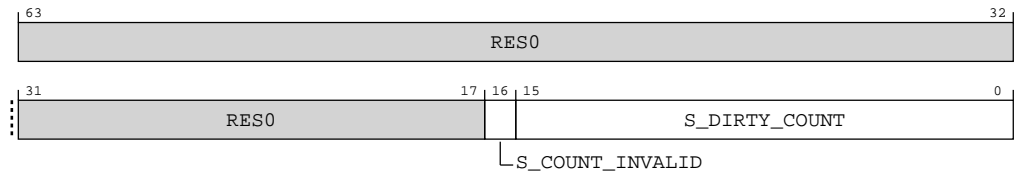
64

Functional group

Generic system control

Reset value

0x0

Bit descriptions**Figure A-31: AArch64_imp_cpul2dirtylnct_el3 bit assignments****Table A-93: IMP_CPUL2SDIRTYLNCT_EL3 bit descriptions**

| Bits | Name | Description | Reset |
|---------|-----------------|---|-------|
| [63:17] | RES0 | Reserved | 0x0 |
| [16] | S_COUNT_INVALID | Indicates the secure dirty count is invalid. Reset value is 'b0 | 0x0 |
| [15:0] | S_DIRTY_COUNT | Number of dirty secure lines in the L2. Reset value is 'h0000 | 0x0 |

Access

MRS <Xt>, S3_6_C15_C2_3

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_3 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b011 |

Accessibility

MRS <Xt>, S3_6_C15_C2_3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUL2SDIRTYLNCT_EL3;

```

A.1.32 IMP_CPUPDPTUNE_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

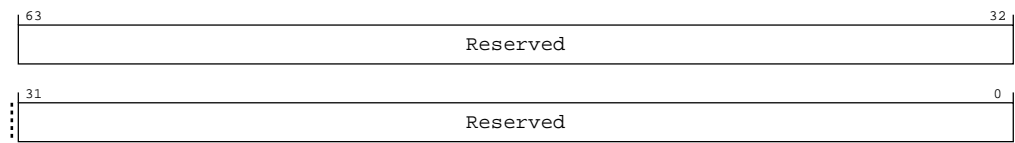
Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-32: AArch64_imp_cpupdtune_el3 bit assignments**Table A-95: IMP_CPUPDPTUNE_EL3 bit descriptions**

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

Access

MRS <Xt>, S3_6_C15_C2_4

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_4 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b100 |

MSR S3_6_C15_C2_4, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_4 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b100 |

Accessibility

MRS <Xt>, S3_6_C15_C2_4

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPDPTUNE_EL3;

```

MSR S3_6_C15_C2_4, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPDPTUNE_EL3 = X[t];
```

A.1.33 IMP_CPUPPMCR4_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-33: AArch64_imp_cpuppmcr4_el3 bit assignments

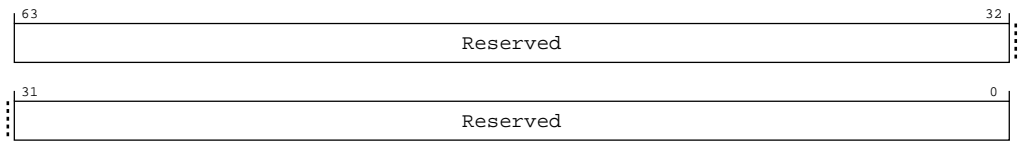


Table A-98: IMP_CPUPPMCR4_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_4

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_4 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b100 |

MSR S3_6_C15_C2_4, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_4 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b100 |

Accessibility

MRS <Xt>, S3_6_C15_C2_4

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR4_EL3;

```

MSR S3_6_C15_C2_4, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPPMCR4_EL3 = X[t];

```

A.1.34 IMP_CPUPDPTUNE2_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes**Width**

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-34: AArch64_imp_cpupdtune2_el3 bit assignments

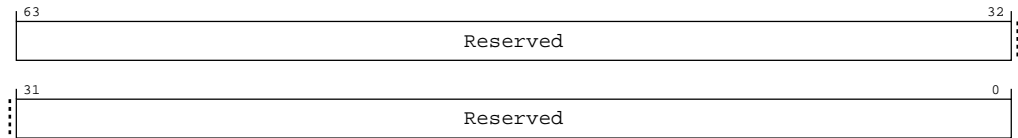


Table A-101: IMP_CPUPDPTUNE2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_5

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_5 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b101 |

MSR S3_6_C15_C2_5, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_5 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b101 |

Accessibility

MRS <Xt>, S3_6_C15_C2_5

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPDPTUNE2_EL3;

```

MSR S3_6_C15_C2_5, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPDPTUNE2_EL3 = X[t];

```

A.1.35 IMP_CPUPPMCR5_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-35: AArch64_imp_cpuppmcr5_el3 bit assignments

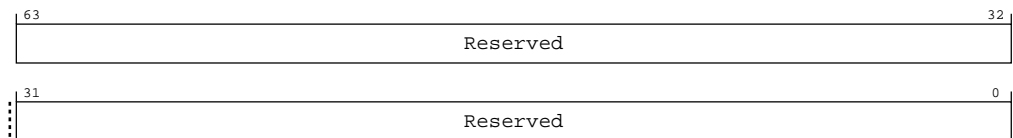


Table A-104: IMP_CPUPPMCR5_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_5

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_5 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b101 |

MSR S3_6_C15_C2_5, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_5 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b101 |

Accessibility

MRS <Xt>, S3_6_C15_C2_5

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

```
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR5_EL3;
```

MSR S3_6_C15_C2_5, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPPMCR5_EL3 = X[t];
```

A.1.36 IMP_CPUMPMMTUNE_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-36: AArch64_imp_cpumpmmtune_el3 bit assignments

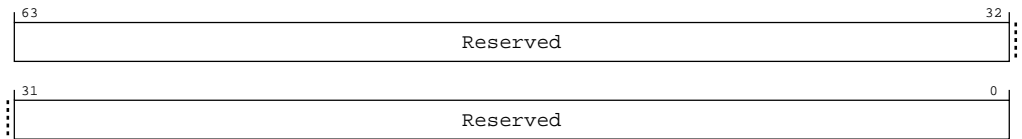


Table A-107: IMP_CPUMPMMTUNE_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_6

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_6 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b110 |

MSR S3_6_C15_C2_6, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_6 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b110 |

Accessibility

MRS <Xt>, S3_6_C15_C2_6

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUMPMTUNE_EL3;

```

MSR S3_6_C15_C2_6, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUMPMTUNE_EL3 = X[t];

```

A.1.37 IMP_CPUPPMCR6_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-37: AArch64_imp_cpuppmcr6_el3 bit assignments

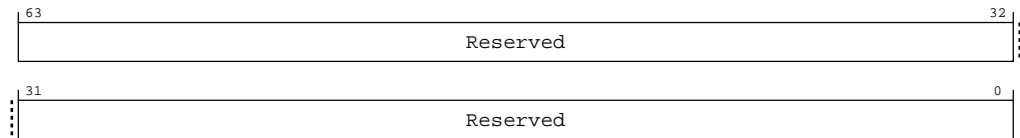


Table A-110: IMP_CPUPPMCR6_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_6

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_6 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b110 |

MSR S3_6_C15_C2_6, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C2_6 | 0b11 | 0b110 | 0b1111 | 0b0010 | 0b110 |

Accessibility

MRS <Xt>, S3_6_C15_C2_6

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR6_EL3;

```

MSR S3_6_C15_C2_6, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPPMCR6_EL3 = X[t];

```

A.1.38 IMP_CPUACTLR_EL3, CPU Auxiliary Control Register (EL3)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-38: AArch64_imp_cpuactlr_el3 bit assignments

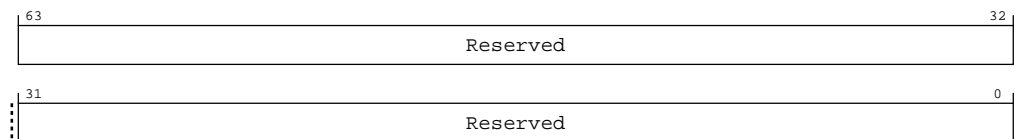


Table A-113: IMP_CPUACTLR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C4_0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C4_0 | 0b11 | 0b110 | 0b1111 | 0b0100 | 0b000 |

MSR S3_6_C15_C4_0, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C4_0 | 0b11 | 0b110 | 0b1111 | 0b0100 | 0b000 |

Accessibility

MRS <Xt>, S3_6_C15_C4_0

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

```

    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR_EL3;

```

MSR S3_6_C15_C4_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUACTLR_EL3 = X[t];

```

A.1.39 IMP_ATCR_EL3, CPU Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-39: AArch64_imp_atcr_el3 bit assignments

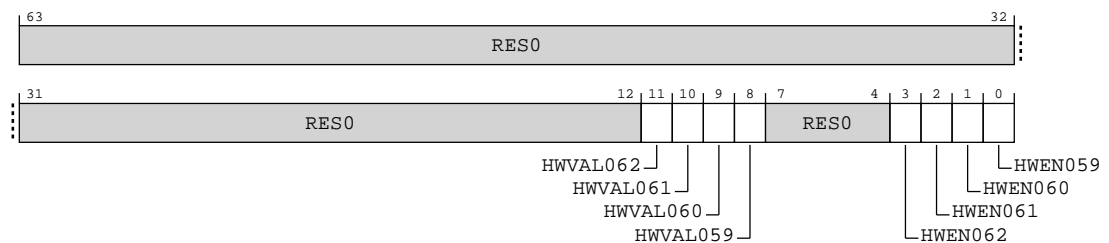


Table A-116: IMP_ATCR_EL3 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|--|--------|
| [63:12] | RES0 | Reserved | 0x0 |
| [11] | HWVAL062 | Value of PBHA[3] on memory accesses due to translation table walks using TTBRO_EL3 if HWEN062 is set. | 0x0 |
| [10] | HWVAL061 | Value of PBHA[2] on memory accesses due to translation table walks using TTBRO_EL3 if HWEN061 is set. | 0x0 |
| [9] | HWVAL060 | Value of PBHA[1] on memory accesses due to translation table walks using TTBRO_EL3 if HWEN060 is set. | 0x0 |
| [8] | HWVAL059 | Value of PBHA[0] on memory accesses due to translation table walks using TTBRO_EL3 if HWEN059 is set. | 0x0 |
| [7:4] | RES0 | Reserved | 0b0000 |
| [3] | HWEN062 | Enable use of PBHA[3] on memory accesses due to translation table walks using TTBRO_EL3. If this bit is clear, PBHA[3] will be 0 on translation table walks. | 0b0 |
| [2] | HWEN061 | Enable use of PBHA[2] on memory accesses due to translation table walks using TTBRO_EL3. If this bit is clear, PBHA[2] will be 0 on translation table walks. | 0b0 |
| [1] | HWEN060 | Enable use of PBHA[1] on memory accesses due to translation table walks using TTBRO_EL3. If this bit is clear, PBHA[1] will be 0 on translation table walks. | 0b0 |
| [0] | HWEN059 | Enable use of PBHA[0] on memory accesses due to translation table walks using TTBRO_EL3. If this bit is clear, PBHA[0] will be 0 on translation table walks. | 0b0 |

Access

MRS <Xt>, S3_6_C15_C7_0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C7_0 | 0b11 | 0b110 | 0b1111 | 0b0111 | 0b000 |

MSR S3_6_C15_C7_0, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C7_0 | 0b11 | 0b110 | 0b1111 | 0b0111 | 0b000 |

Accessibility

MRS <Xt>, S3_6_C15_C7_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_ATCR_EL3;

```

MSR S3_6_C15_C7_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```

```
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_ATCR_EL3 = X[t];
```

A.1.40 IMP_CPUPSELR_EL3, Selected Instruction Private Select Register

Selects the current instruction patch register for subsequent accesses to AArch64-IMP_CPUPCR_EL3, AArch64-IMP_CPUPOR_EL3, AArch64-IMP_CPUPMR_EL3, AArch64-IMP_CPUPOR2_EL3, AArch64-IMP_CPUPMR2_EL3, and AArch64-IMP_CPUPFR_EL3

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-40: AArch64_imp_cpupselr_el3 bit assignments



Table A-119: IMP_CPUPSELR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_0 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b000 |

MSR S3_6_C15_C8_0, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_0 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b000 |

Accessibility

MRS <Xt>, S3_6_C15_C8_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPSELR_EL3;

```

MSR S3_6_C15_C8_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPSELR_EL3 = X[t];

```

A.1.41 IMP_CPUPCR_EL3, Selected Instruction Private Control Register

Configures current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-41: AArch64_imp_cpupcr_el3 bit assignments

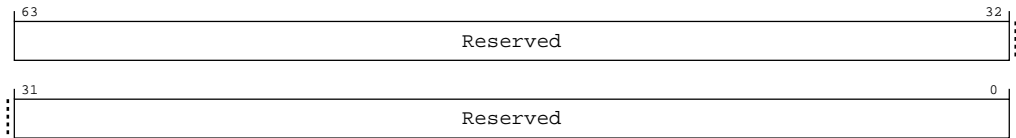


Table A-122: IMP_CPUPCR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_1 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b001 |

MSR S3_6_C15_C8_1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_1 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b001 |

Accessibility

MRS <Xt>, S3_6_C15_C8_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPCR_EL3;

```

MSR S3_6_C15_C8_1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPCR_EL3 = X[t];

```

A.1.42 IMP_CPUPOR_EL3, Selected Instruction Private Opcode Register

Opcode for current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-42: AArch64_imp_cpupor_el3 bit assignments

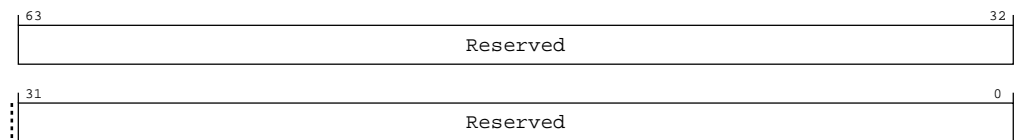


Table A-125: IMP_CPUPOR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_2 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b010 |

MSR S3_6_C15_C8_2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_2 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b010 |

Accessibility

MRS <Xt>, S3_6_C15_C8_2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```

```
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPOR_EL3;
```

MSR S3_6_C15_C8_2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPOR_EL3 = X[t];
```

A.1.43 IMP_CPUPMR_EL3, Selected Instruction Private Mask Register

Mask for current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-43: AArch64_imp_cpupmr_el3 bit assignments

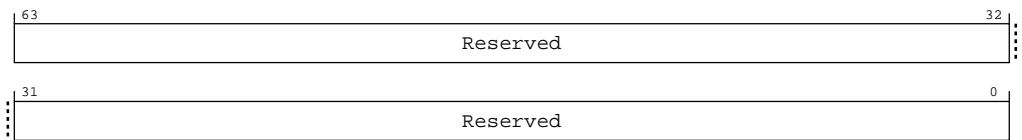


Table A-128: IMP_CPUPMR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_3

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_3 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b011 |

MSR S3_6_C15_C8_3, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_3 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b011 |

Accessibility

MRS <Xt>, S3_6_C15_C8_3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPMR_EL3;
```

MSR S3_6_C15_C8_3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPMR_EL3 = X[t];
```

A.1.44 IMP_CPUPOR2_EL3, Selected Instruction Private Opcode Register 2

Opcode exclusion for current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-44: AArch64_imp_cpupor2_el3 bit assignments

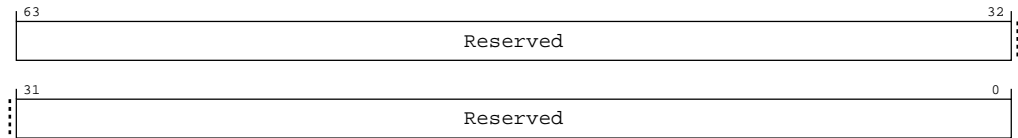


Table A-131: IMP_CPUPOR2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_4

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_4 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b100 |

MSR S3_6_C15_C8_4, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_4 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b100 |

Accessibility

MRS <Xt>, S3_6_C15_C8_4

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPOR2_EL3;

```

MSR S3_6_C15_C8_4, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPOR2_EL3 = X[t];

```

A.1.45 IMP_CPUPMR2_EL3, Selected Instruction Private Mask Register 2

Mask exclusion for current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-45: AArch64_imp_cpupmr2_el3 bit assignments

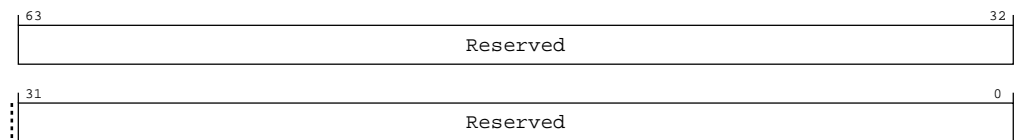


Table A-134: IMP_CPUPMR2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_5

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_5 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b101 |

MSR S3_6_C15_C8_5, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_5 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b101 |

Accessibility

MRS <Xt>, S3_6_C15_C8_5

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```

```
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPMR2_EL3;
```

MSR S3_6_C15_C8_5, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPMR2_EL3 = X[t];
```

A.1.46 IMP_CPUPFR_EL3, Selected Instruction Private Flag Register

Instruction Patch flags for current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-46: AArch64_imp_cpupfr_el3 bit assignments

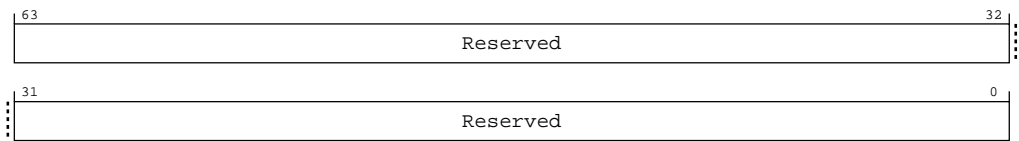


Table A-137: IMP_CPUPFR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_6

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_6 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b110 |

MSR S3_6_C15_C8_6, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C8_6 | 0b11 | 0b110 | 0b1111 | 0b1000 | 0b110 |

Accessibility

MRS <Xt>, S3_6_C15_C8_6

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPFR_EL3;

```

MSR S3_6_C15_C8_6, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPFR_EL3 = X[t];

```

A.1.47 FPCR, Floating-point Control Register

Controls floating-point behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets

Bit descriptions

Figure A-47: AArch64_fpcr bit assignments

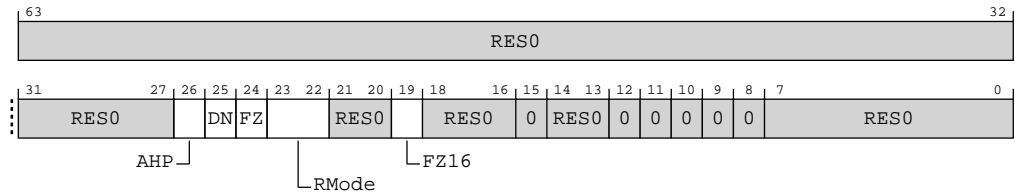


Table A-140: FPCR bit descriptions

| Bits | Name | Description | Reset |
|---------|------|---|-------|
| [63:27] | RES0 | Reserved | 0x0 |
| [26] | AHP | <p>Alternative half-precision control bit:</p> <p>0b0 IEEE half-precision format selected.</p> <p>0b1 Alternative half-precision format selected.</p> <p>This bit is only used for conversions between half-precision floating-point and other floating-point formats.</p> <p>The data-processing instructions added as part of the Armv8.2-FP16 extension always use the IEEE half-precision format, and ignore the value of this bit.</p> | |
| [25] | DN | <p>Default NaN mode control bit:</p> <p>0b0 NaN operands propagate through to the output of a floating-point operation.</p> <p>0b1 Any operation involving one or more NaNs returns the Default NaN.</p> <p>The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.</p> | |
| [24] | FZ | <p>Flush-to-zero mode control bit.</p> <p>0b0 Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.</p> <p>0b1 Flush-to-zero mode enabled.</p> <p>The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.</p> <p>This bit has no effect on half-precision calculations.</p> | |

| Bits | Name | Description | Reset |
|---------|-------|--|------------|
| [23:22] | RMode | <p>Rounding Mode control field.</p> <p>0b00 Round to Nearest (RN) mode.</p> <p>0b01 Round towards Plus Infinity (RP) mode.</p> <p>0b10 Round towards Minus Infinity (RM) mode.</p> <p>0b11 Round towards Zero (RZ) mode.</p> <p>The specified rounding mode is used by both scalar and Advanced SIMD floating-point instructions.</p> | |
| [21:20] | RES0 | Reserved | 0b00 |
| [19] | FZ16 | <p>Flush-to-zero mode control bit on half-precision data-processing instructions.</p> <p>0b0 Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.</p> <p>0b1 Flush-to-zero mode enabled.</p> <p>The value of this bit applies to both scalar and Advanced SIMD floating-point half-precision calculations. A half-precision floating-point number that is flushed to zero as a result of the value of the FZ16 bit does not generate an Input Denormal exception.</p> | |
| [18:0] | RES0 | Reserved | 0b00000000 |

Access

MRS <Xt>, FPCR

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| FPCR | 0b11 | 0b011 | 0b0100 | 0b0100 | 0b000 |

MSR FPCR, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| FPCR | 0b11 | 0b011 | 0b0100 | 0b0100 | 0b000 |

Accessibility

MRS <Xt>, FPCR

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
        elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.FPEN != '11' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elseif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x07);

```

```

    elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        return FPCR;
    elsif PSTATE.EL == EL1 then
        if CPACR_EL1.FPEN == 'x0' then
            AArch64.SystemAccessTrap(EL1, 0x07);
        elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elsif CPTR_EL3.TFP == '1' then
            AArch64.SystemAccessTrap(EL3, 0x07);
        else
            return FPCR;
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elsif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elsif CPTR_EL3.TFP == '1' then
            AArch64.SystemAccessTrap(EL3, 0x07);
        else
            return FPCR;
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TFP == '1' then
            AArch64.SystemAccessTrap(EL3, 0x07);
        else
            return FPCR;

```

MSR FPCR, <Xt>

```

    if PSTATE.EL == EL0 then
        if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.FPEN != '11' then
            if EL2Enabled() && HCR_EL2.TGE == '1' then
                AArch64.SystemAccessTrap(EL2, 0x00);
            else
                AArch64.SystemAccessTrap(EL1, 0x07);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.FPEN != '11' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elsif CPTR_EL3.TFP == '1' then
            AArch64.SystemAccessTrap(EL3, 0x07);
        else
            FPCR = X[t];
    elsif PSTATE.EL == EL1 then
        if CPACR_EL1.FPEN == 'x0' then
            AArch64.SystemAccessTrap(EL1, 0x07);
        elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elsif CPTR_EL3.TFP == '1' then
            AArch64.SystemAccessTrap(EL3, 0x07);
        else
            FPCR = X[t];
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elsif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        elsif CPTR_EL3.TFP == '1' then
            AArch64.SystemAccessTrap(EL3, 0x07);

```

```
    else
        FPCR = X[t];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TFP == '1' then
            AArch64.SystemAccessTrap(EL3, 0x07);
        else
            FPCR = X[t];
        end if
    end if
```

A.1.48 AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2)

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

Configurations

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

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-48: AArch64_afsr0_el2 bit assignments

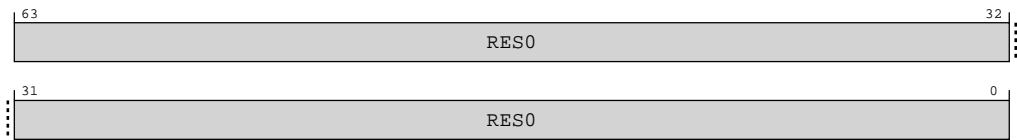


Table A-143: AFSR0_EL2 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, AFSR0_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSR0_EL2 | 0b11 | 0b100 | 0b0101 | 0b0001 | 0b000 |

MSR AFSR0_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSRO_EL2 | 0b11 | 0b100 | 0b0101 | 0b0001 | 0b000 |

MRS <Xt>, AFSRO_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSRO_EL1 | 0b11 | 0b000 | 0b0101 | 0b0001 | 0b000 |

MSR AFSRO_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSRO_EL1 | 0b11 | 0b000 | 0b0101 | 0b0001 | 0b000 |

Accessibility

MRS <Xt>, AFSRO_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    return AFSRO_EL2;
elseif PSTATE.EL == EL3 then
    return AFSRO_EL2;

```

MSR AFSRO_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AFSRO_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    AFSRO_EL2 = X[t];

```

MRS <Xt>, AFSRO_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AFSRO_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x128];
    else
        return AFSRO_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then

```

```

        return AFSR0_EL2;
    else
        return AFSR0_EL1;
    elsif PSTATE.EL == EL3 then
        return AFSR0_EL1;

```

MSR AFSR0_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x128] = X[t];
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL2 = X[t];
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t];

```

A.1.49 AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2)

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

Configurations

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

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-49: AArch64_afsr1_el2 bit assignments

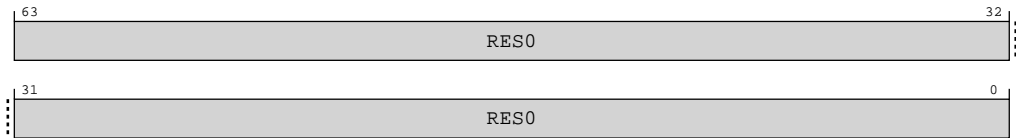


Table A-148: AFSR1_EL2 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, AFSR1_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSR1_EL2 | 0b11 | 0b100 | 0b0101 | 0b0001 | 0b001 |

MSR AFSR1_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSR1_EL2 | 0b11 | 0b100 | 0b0101 | 0b0001 | 0b001 |

MRS <Xt>, AFSR1_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSR1_EL1 | 0b11 | 0b000 | 0b0101 | 0b0001 | 0b001 |

MSR AFSR1_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSR1_EL1 | 0b11 | 0b000 | 0b0101 | 0b0001 | 0b001 |

Accessibility

MRS <Xt>, AFSR1_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return AFSR1_EL2;
elsif PSTATE.EL == EL3 then
    return AFSR1_EL2;

```

MSR AFSR1_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AFSR1_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    AFSR1_EL2 = X[t];

```

MRS <Xt>, AFSR1_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x130];
    else
        return AFSR1_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR1_EL2;
    else
        return AFSR1_EL1;
elseif PSTATE.EL == EL3 then
    return AFSR1_EL1;

```

MSR AFSR1_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x130] = X[t];
    else
        AFSR1_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL2 = X[t];
    else
        AFSR1_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    AFSR1_EL1 = X[t];

```

A.1.50 AFSRO_EL1, Auxiliary Fault Status Register 0 (EL1)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-50: AArch64_afsr0_el1 bit assignments

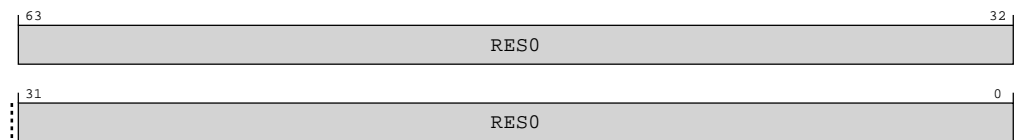


Table A-153: AFSRO_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, AFSRO_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSRO_EL1 | 0b11 | 0b000 | 0b0101 | 0b0001 | 0b000 |

MSR AFSRO_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSRO_EL1 | 0b11 | 0b000 | 0b0101 | 0b0001 | 0b000 |

MRS <Xt>, AFSRO_EL12

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSRO_EL12 | 0b11 | 0b101 | 0b0101 | 0b0001 | 0b000 |

MSR AFSRO_EL12, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSRO_EL12 | 0b11 | 0b101 | 0b0101 | 0b0001 | 0b000 |

Accessibility

MRS <Xt>, AFSRO_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AFSRO_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x128];
    else
        return AFSRO_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSRO_EL2;
    else
        return AFSRO_EL1;
elseif PSTATE.EL == EL3 then
    return AFSRO_EL1;

```

MSR AFSRO_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSRO_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x128] = X[t];
    else
        AFSRO_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSRO_EL2 = X[t];
    else
        AFSRO_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    AFSRO_EL1 = X[t];

```

MRS <Xt>, AFSRO_EL12

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        return NVMem[0x128];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then

```

```

        return AFSR0_EL1;
    else
        UNDEFINED;
elseif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        return AFSR0_EL1;
    else
        UNDEFINED;

```

MSR AFSR0_EL12, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x128] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL1 = X[t];
    else
        UNDEFINED;
elseif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        AFSR0_EL1 = X[t];
    else
        UNDEFINED;

```

A.1.51 AFSR1_EL1, Auxiliary Fault Status Register 1 (EL1)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-51: AArch64_afsr1_el1 bit assignments

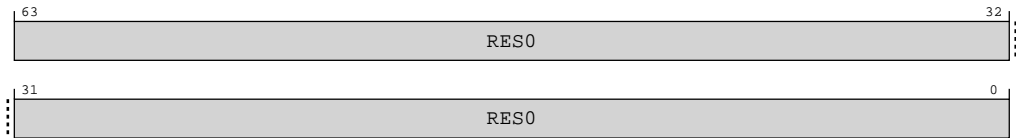


Table A-158: AFSR1_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, AFSR1_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSR1_EL1 | 0b11 | 0b000 | 0b0101 | 0b0001 | 0b001 |

MSR AFSR1_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSR1_EL1 | 0b11 | 0b000 | 0b0101 | 0b0001 | 0b001 |

MRS <Xt>, AFSR1_EL12

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

MSR AFSR1_EL12, <Xt>

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

Accessibility

MRS <Xt>, AFSR1_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x130];
    else
        return AFSR1_EL1;
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then

```

```

        return AFSR1_EL2;
    else
        return AFSR1_EL1;
    elsif PSTATE.EL == EL3 then
        return AFSR1_EL1;

```

MSR AFSR1_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x130] = X[t];
    else
        AFSR1_EL1 = X[t];
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            AFSR1_EL2 = X[t];
        else
            AFSR1_EL1 = X[t];
    elsif PSTATE.EL == EL3 then
        AFSR1_EL1 = X[t];

```

MRS <Xt>, AFSR1_EL12

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        return NVMem[0x130];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            return AFSR1_EL1;
        else
            UNDEFINED;
    elsif PSTATE.EL == EL3 then
        if EL2Enabled() && HCR_EL2.E2H == '1' then
            return AFSR1_EL1;
        else
            UNDEFINED;

```

MSR AFSR1_EL12, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x130] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            AFSR1_EL1 = X[t];

```

```
    else
        UNDEFINED;
    elsif PSTATE.EL == EL3 then
        if EL2Enabled() && HCR_EL2.E2H == '1' then
            AFSR1_EL1 = X[t];
        else
            UNDEFINED;
```

A.1.52 AFSR0_EL3, Auxiliary Fault Status Register 0 (EL3)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-52: AArch64_afsr0_el3 bit assignments

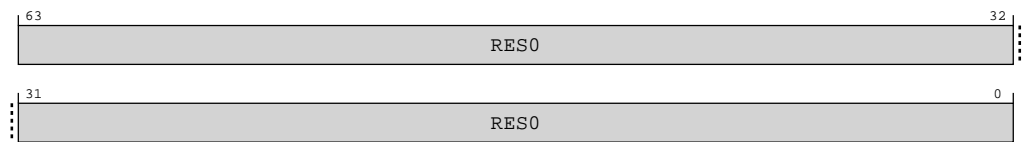


Table A-163: AFSR0_EL3 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, AFSR0_EL3

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

MSR AFSR0_EL3, <Xt>

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

Accessibility

MRS <Xt>, AFSR0_EL3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return AFSR0_EL3;

```

MSR AFSR0_EL3, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    AFSR0_EL3 = X[t];

```

A.1.53 AFSR1_EL3, Auxiliary Fault Status Register 1 (EL3)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-53: AArch64_afsr1_el3 bit assignments

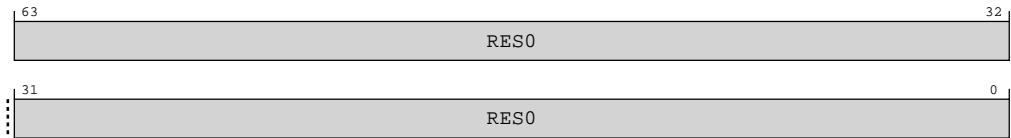


Table A-166: AFSR1_EL3 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, AFSR1_EL3

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSR1_EL3 | 0b11 | 0b110 | 0b0101 | 0b0001 | 0b001 |

MSR AFSR1_EL3, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AFSR1_EL3 | 0b11 | 0b110 | 0b0101 | 0b0001 | 0b001 |

Accessibility

MRS <Xt>, AFSR1_EL3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return AFSR1_EL3;

```

MSR AFSR1_EL3, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    AFSR1_EL3 = X[t];

```

A.2 AArch64 Debug register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Debug registers in the core. For more information about a register, you can click the register name in the table.

Table A-169: AArch64 Debug register summary

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

A.2.1 IMP_IDATA0_EL3, Instruction Register 0

Contains data from a preceding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

Reset value

See individual bit resets

Bit descriptions

Figure A-54: AArch64_imp_idata0_el3 bit assignments

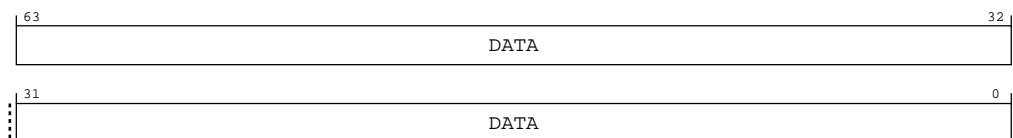


Table A-170: IMP_IDATA0_EL3 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|---|-------|
| [63:0] | DATA | Contains data from a preceding RAMINDEX operation | |

Access

MRS <Xt>, S3_6_C15_C0_0

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

Accessibility

MRS <Xt>, S3_6_C15_C0_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_IDATA0_EL3;

```

A.2.2 IMP_IDATA1_EL3, Instruction Register 0

Contains data from a preceding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

Reset value

See individual bit resets

Bit descriptions

Figure A-55: AArch64_imp_idata1_el3 bit assignments

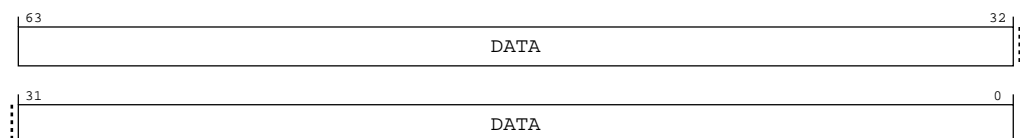


Table A-172: IMP_IDATA1_EL3 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|---|-------|
| [63:0] | DATA | Contains data from a preceding RAMINDEX operation | |

Access

MRS <Xt>, S3_6_C15_CO_1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_CO_1 | 0b11 | 0b110 | 0b1111 | 0b0000 | 0b001 |

Accessibility

MRS <Xt>, S3_6_C15_CO_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_IDATA1_EL3;

```

A.2.3 IMP_IDATA2_EL3, Instruction Register 0

Contains data from a preceding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

Reset value

See individual bit resets

Bit descriptions

Figure A-56: AArch64_imp_idata2_el3 bit assignments

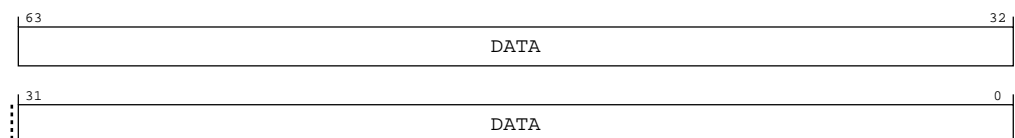


Table A-174: IMP_IDATA2_EL3 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|---|-------|
| [63:0] | DATA | Contains data from a preceding RAMINDEX operation | |

Access

MRS <Xt>, S3_6_C15_C0_2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C0_2 | 0b11 | 0b110 | 0b1111 | 0b0000 | 0b010 |

Accessibility

MRS <Xt>, S3_6_C15_C0_2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_IDATA2_EL3;

```

A.2.4 IMP_DDATA0_EL3, Data Register 0

Contains data from a preceding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

Reset value

See individual bit resets

Bit descriptions

Figure A-57: AArch64_imp_ddata0_el3 bit assignments

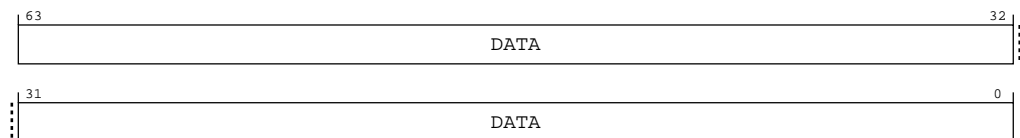


Table A-176: IMP_DDATA0_EL3 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|---|-------|
| [63:0] | DATA | Contains data from a preceding RAMINDEX operation | |

Access

MRS <Xt>, S3_6_C15_C1_0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C1_0 | 0b11 | 0b110 | 0b1111 | 0b0001 | 0b000 |

Accessibility

MRS <Xt>, S3_6_C15_C1_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_DDATA0_EL3;

```

A.2.5 IMP_DDATA1_EL3, Data Register 1

Contains data from a preceding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

Reset value

See individual bit resets

Bit descriptions

Figure A-58: AArch64_imp_ddata1_el3 bit assignments

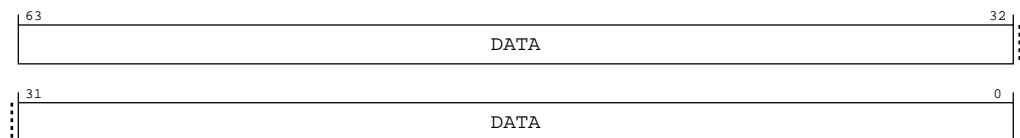


Table A-178: IMP_DDATA1_EL3 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|---|-------|
| [63:0] | DATA | Contains data from a preceding RAMINDEX operation | |

Access

MRS <Xt>, S3_6_C15_C1_1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C1_1 | 0b11 | 0b110 | 0b1111 | 0b0001 | 0b001 |

Accessibility

MRS <Xt>, S3_6_C15_C1_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_DDATA1_EL3;

```

A.2.6 IMP_DDATA2_EL3, Data Register 2

Contains data from a preceding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

Reset value

See individual bit resets

Bit descriptions

Figure A-59: AArch64_imp_ddata2_el3 bit assignments

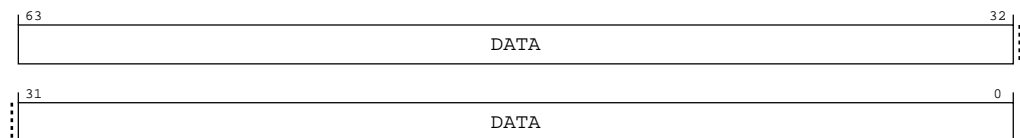


Table A-180: IMP_DDATA2_EL3 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|---|-------|
| [63:0] | DATA | Contains data from a preceding RAMINDEX operation | |

Access

MRS <Xt>, S3_6_C15_C1_2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C1_2 | 0b11 | 0b110 | 0b1111 | 0b0001 | 0b010 |

Accessibility

MRS <Xt>, S3_6_C15_C1_2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_DDATA2_EL3;

```

A.3 AArch64 Random number control register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** random number control registers in the core. For more information about a register, you can click the register name in the table.

Table A-182: AArch64 Random number control register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|------------------------------------|-----|-----|-----|-----|-----|-------|--------|--|
| IMP_CPURNDBR_EL3 | 3 | C15 | 6 | C3 | 0 | 0x0 | 64-bit | CPU Random Number Base Register |
| IMP_CPURNDPEID_EL3 | 3 | C15 | 6 | C3 | 1 | 0x0 | 64-bit | CPU Random Number Packet Identification Register |

A.3.1 IMP_CPURNDBR_EL3, CPU Random Number Base Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Random number control

Reset value

0x0

Bit descriptions

Figure A-60: AArch64_imp_cpurndb_r_el3 bit assignments

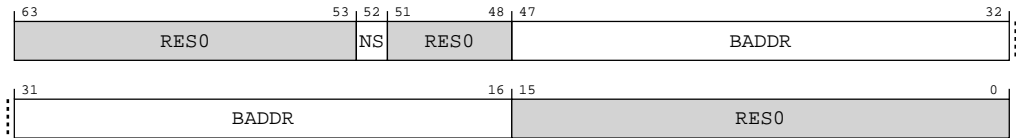


Table A-183: IMP_CPURNDBR_EL3 bit descriptions

| Bits | Name | Description | Reset |
|---------|-------|---|--------|
| [63:53] | RES0 | Reserved | 0x0 |
| [52] | NS | Indicates the security state of the external RNG block accesses. The possible values are: 0b0 Secure 0b1 Non-secure | 0x0 |
| [51:48] | RES0 | Reserved | 0b0000 |
| [47:16] | BADDR | Indicates the base address bits [47:16] of the external RNG block | 0x0 |
| [15:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, S3_6_C15_C3_0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C3_0 | 0b11 | 0b110 | 0b1111 | 0b0011 | 0b000 |

MSR S3_6_C15_C3_0, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C3_0 | 0b11 | 0b110 | 0b1111 | 0b0011 | 0b000 |

Accessibility

MRS <Xt>, S3_6_C15_C3_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPURNDBR_EL3;

```

MSR S3_6_C15_C3_0, <Xt>

```

if PSTATE.EL == EL0 then

```

```

    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPURNDBR_EL3 = X[t];

```

A.3.2 IMP_CPURNDPEID_EL3, CPU Random Number Packet Identification Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Random number control

Reset value

0x0

Bit descriptions

Figure A-61: AArch64_imp_cpurndpeid_el3 bit assignments

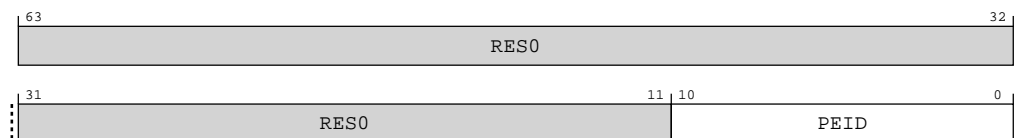


Table A-186: IMP_CPURNDPEID_EL3 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|--|-------|
| [63:11] | RES0 | Reserved | 0x0 |
| [10:0] | PEID | Unique 11-bit hardware identification which is used to construct the address for RNDR accesses: RNDR address={CPURNDBR_EL3[47:16],CPURNDPEID_EL3[10:0],1'b0,4'b0}, RNDRRS address={CPURNDBR_EL3[47:16],CPURNDPEID_EL3[10:0],1'b1,4'b0} | 0x0 |

Access

MRS <Xt>, S3_6_C15_C3_1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C3_1 | 0b11 | 0b110 | 0b1111 | 0b0011 | 0b001 |

MSR S3_6_C15_C3_1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_6_C15_C3_1 | 0b11 | 0b110 | 0b1111 | 0b0011 | 0b001 |

Accessibility

MRS <Xt>, S3_6_C15_C3_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPURNDPEID_EL3;

```

MSR S3_6_C15_C3_1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPURNDPEID_EL3 = X[t];

```

A.4 AArch64 System instruction register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** system instruction registers in the core. For more information about a register, you can click the register name in the table.

Table A-189: AArch64 System instruction register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|----------------------------------|-----|-----|-----|-----|-----|---------------------------|--------|-------------|
| SYS_IMP_RAMINDEX | 1 | C15 | 6 | C0 | 0 | See individual bit resets | 64-bit | RAM Index |

A.4.1 SYS_IMP_RAMINDEX, RAM Index

Read contents of the cache specified by the source register into AArch64-IMP_IDATA0_EL3, AArch64-IMP_IDATA1_EL3, AArch64-IMP_IDATA2_EL3, AArch64-IMP_DDATA0_EL3, AArch64-IMP_DDATA1_EL3, and AArch64-IMP_DDATA2_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

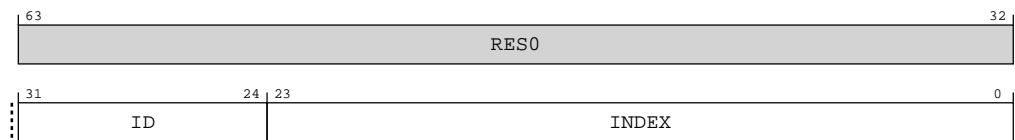
Functional group

System instruction

Reset value

See individual bit resets

Bit descriptions

Figure A-62: AArch64_sys_imp_ramindex bit assignments**Table A-190: SYS_IMP_RAMINDEX bit descriptions**

| Bits | Name | Description | Reset |
|---------|-------|----------------------------|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:24] | ID | RAM ID (See Chapter 10) | |
| [23:0] | INDEX | RAM Index (See Chapter 10) | |

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C0, #0, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S1_6_C15_C0_0 | 0b01 | 0b110 | 0b1111 | 0b0000 | 0b000 |

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C0, #0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_RAMINDEX(X[t]);

```

A.5 AArch64 Identification register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** identification registers in the core. For more information about a register, you can click the register name in the table.

Table A-192: AArch64 Identification register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|----------------------------------|-----|-----|-----|-----|-----|---------------------------|--------|--|
| MIDR_EL1 | 3 | C0 | 0 | C0 | 0 | See individual bit resets | 64-bit | Main ID Register |
| MPIDR_EL1 | 3 | C0 | 0 | C0 | 5 | See individual bit resets | 64-bit | Multiprocessor Affinity Register |
| REVIDR_EL1 | 3 | C0 | 0 | C0 | 6 | See individual bit resets | 64-bit | Revision ID Register |
| ID_AA64PFR0_EL1 | 3 | C0 | 0 | C4 | 0 | See individual bit resets | 64-bit | AArch64 Processor Feature Register 0 |
| ID_AA64PFR1_EL1 | 3 | C0 | 0 | C4 | 1 | See individual bit resets | 64-bit | AArch64 Processor Feature Register 1 |
| ID_AA64ZFR0_EL1 | 3 | C0 | 0 | C4 | 4 | See individual bit resets | 64-bit | SVE Feature ID register 0 |
| ID_AA64DFR0_EL1 | 3 | C0 | 0 | C5 | 0 | See individual bit resets | 64-bit | AArch64 Debug Feature Register 0 |
| ID_AA64DFR1_EL1 | 3 | C0 | 0 | C5 | 1 | 0x0 | 64-bit | AArch64 Debug Feature Register 1 |
| ID_AA64AFR0_EL1 | 3 | C0 | 0 | C5 | 4 | 0x0 | 64-bit | AArch64 Auxiliary Feature Register 0 |
| ID_AA64AFR1_EL1 | 3 | C0 | 0 | C5 | 5 | 0x0 | 64-bit | AArch64 Auxiliary Feature Register 1 |
| ID_AA64ISAR0_EL1 | 3 | C0 | 0 | C6 | 0 | See individual bit resets | 64-bit | AArch64 Instruction Set Attribute Register 0 |
| ID_AA64ISAR1_EL1 | 3 | C0 | 0 | C6 | 1 | See individual bit resets | 64-bit | AArch64 Instruction Set Attribute Register 1 |
| ID_AA64MMFR0_EL1 | 3 | C0 | 0 | C7 | 0 | See individual bit resets | 64-bit | AArch64 Memory Model Feature Register 0 |
| ID_AA64MMFR1_EL1 | 3 | C0 | 0 | C7 | 1 | See individual bit resets | 64-bit | AArch64 Memory Model Feature Register 1 |
| ID_AA64MMFR2_EL1 | 3 | C0 | 0 | C7 | 2 | See individual bit resets | 64-bit | AArch64 Memory Model Feature Register 2 |
| CLIDR_EL1 | 3 | C0 | 1 | C0 | 1 | See individual bit resets | 64-bit | Cache Level ID Register |
| GMIDR_EL1 | 3 | C0 | 1 | C0 | 4 | See individual bit resets | 64-bit | Multiple tag transfer ID register |
| CTR_EL0 | 3 | C0 | 3 | C0 | 1 | See individual bit resets | 64-bit | Cache Type Register |
| DCZID_EL0 | 3 | C0 | 3 | C0 | 7 | See individual bit resets | 64-bit | Data Cache Zero ID register |
| MPAMIDR_EL1 | 3 | C10 | 0 | C4 | 4 | See individual bit resets | 64-bit | MPAM ID Register (EL1) |
| IMP_CPUCFR_EL1 | 3 | C15 | 0 | C0 | 0 | See individual bit resets | 64-bit | CPU Configuration Register |

A.5.1 MIDR_EL1, Main ID Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-63: AArch64_midr_el1 bit assignments

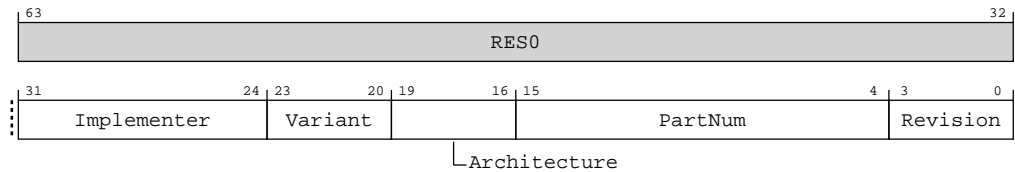


Table A-193: MIDR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|--------------|--|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:24] | Implementer | Indicates the implementer code. This value is: 0b01000001 Arm Limited | |
| [23:20] | Variant | An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product. 0b0000 rOp1 | |
| [19:16] | Architecture | Indicates the architecture code. This value is: 0b1111 Architecture is defined by ID registers | |
| [15:4] | PartNum | An IMPLEMENTATION DEFINED primary part number for the device. On processors implemented by Arm, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently. 0b110101001111 Neoverse™ V2 | |
| [3:0] | Revision | An IMPLEMENTATION DEFINED revision number for the device. 0b0001 rOp1 | |

Access

MRS <Xt>, MIDR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| MIDR_EL1 | 0b11 | 0b000 | 0b0000 | 0b0000 | 0b000 |

Accessibility

MRS <Xt>, MIDR_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.MIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() then
        return VPIDR_EL2;
    else
        return MIDR_EL1;
elseif PSTATE.EL == EL2 then
    return MIDR_EL1;
elseif PSTATE.EL == EL3 then
    return MIDR_EL1;

```

A.5.2 MPIDR_EL1, Multiprocessor Affinity Register

In a multiprocessor system, provides an additional PE identification mechanism for scheduling purposes.

Configurations

In a uniprocessor system Arm recommends that each Aff<n> field of this register returns a value of 0.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-64: AArch64_mpidr_el1 bit assignments

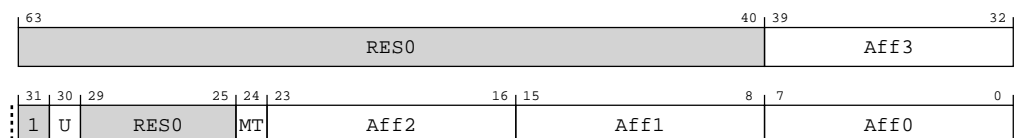


Table A-195: MPIDR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|---|---------|
| [63:40] | RES0 | Reserved | 0x0 |
| [39:32] | Aff3 | Affinity level 3. See the description of Aff0 for more information. The value will be determined by the CLUSTERIDAFF3 configuration pins. | |
| [31] | RES1 | Reserved | 0b1 |
| [30] | U | Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system. The possible values of this bit are: 0b0 Processor is part of a multiprocessor system. | |
| [29:25] | RES0 | Reserved | 0b00000 |
| [24] | MT | Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. See the description of Aff0 for more information about affinity levels. The possible values of this bit are: 0b1 Performance of PEs at the lowest affinity level, or PEs with MPIDR_EL1.MT set to 1, different affinity level 0 values, and the same values for affinity level 1 and higher, is very interdependent. | |
| [23:16] | Aff2 | Affinity level 2. See the description of Aff0 for more information. The value will be determined by the CLUSTERIDAFF2 configuration pins. | |
| [15:8] | Aff1 | Affinity level 1. See the description of Aff0 for more information. Value read from the CUID configuration pins. Identification number for each CPU in an cluster counting from zero. | |
| [7:0] | Aff0 | Affinity level 0. This is the affinity level that is most significant for determining PE behavior. Higher affinity levels are increasingly less significant in determining PE behavior. The assigned value of the MPIDR.{Aff2, Aff1, Aff0} or AArch64-MPIDR_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole. 0b00000000 Only one thread. | |

Access

MRS <Xt>, MPIDR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| MPIDR_EL1 | 0b11 | 0b000 | 0b0000 | 0b0000 | 0b101 |

Accessibility

MRS <Xt>, MPIDR_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.MPIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);

```

```

    elif EL2Enabled() then
        return VMPIDR_EL2;
    else
        return MPIDR_EL1;
    elif PSTATE.EL == EL2 then
        return MPIDR_EL1;
    elif PSTATE.EL == EL3 then
        return MPIDR_EL1;

```

A.5.3 REVIDR_EL1, Revision ID Register

The REVIDR_EL1 provides revision information, additional to MIDR_EL1, that identifies minor fixes (errata) which might be present in a specific implementation of the Neoverse™ V2 core. Refer to the Neoverse™ V2 Core Product Errata Notice (PEN) for information on how to interpret the values in this register.

Configurations

If REVIDR_EL1 has the same value as AArch64-MIDR_EL1, then its contents have no significance.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-65: AArch64_revidr_el1 bit assignments

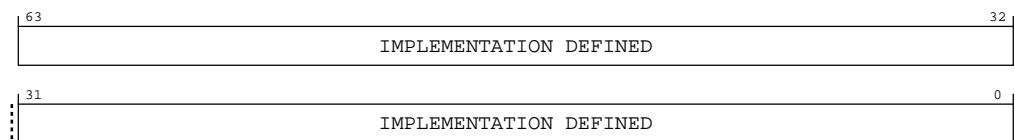


Table A-197: REVIDR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------------------------|------------------------|-------|
| [63:0] | IMPLEMENTATION DEFINED | IMPLEMENTATION DEFINED | |

Access

MRS <Xt>, REVIDR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| REVIDR_EL1 | 0b11 | 0b000 | 0b0000 | 0b0000 | 0b110 |

Accessibility

MRS <Xt>, REVIDR_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.REVIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return REVIDR_EL1;
elseif PSTATE.EL == EL2 then
    return REVIDR_EL1;
elseif PSTATE.EL == EL3 then
    return REVIDR_EL1;

```

A.5.4 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0

Provides additional information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

The external register ext-EDPFR gives information from this register.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-66: AArch64_id_aa64pfr0_el1 bit assignments

| | | | | | | | | | | | | | | | |
|------|------|---------|-----|-----|------|------|-----|----|----|----|----|----|----|----|----|
| 63 | 60 | 59 | 56 | 55 | 52 | 51 | 48 | 47 | 44 | 43 | 40 | 39 | 36 | 35 | 32 |
| CSV3 | CSV2 | RES0 | DIT | AMU | MPAM | SEL2 | SVE | | | | | | | | |
| 31 | 28 | 27 | 24 | 23 | 20 | 19 | 16 | 15 | 12 | 11 | 8 | 7 | 4 | 3 | 0 |
| RAS | GIC | AdvSIMD | FP | EL3 | EL2 | EL1 | EL0 | | | | | | | | |

Table A-199: ID_AA64PFR0_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|---|--------|
| [63:60] | CSV3 | Speculative use of faulting data. Defined values are: 0b0001 Data loaded under speculation with a permission or domain fault cannot be used to form an address or generate condition codes or SVE predicate values to be used by instructions newer than the load in the speculative sequence | |
| [59:56] | CSV2 | Speculative use of out of context branch targets. Defined values are: 0b0010 Branch targets trained in one hardware described context can only affect speculative execution in a different hardware described context in a hard-to-determine way. Contexts include the SCXTNUM_ELx register contexts, and these registers are supported. | |
| [55:52] | RES0 | Reserved | 0b0000 |
| [51:48] | DIT | Data Independent Timing. Defined values are: 0b0001 AArch64 provides the PSTATE.DIT mechanism to guarantee constant execution time of certain instructions. | |
| [47:44] | AMU | Indicates support for Activity Monitors Extension. Defined values are: 0b0001 AMUv1 for Armv8.4 is implemented. | |
| [43:40] | MPAM | Indicates support for MPAM Extension. Defined values are: 0b0001 If AArch64-ID_AA64PFR1_EL1.MPAM_frac == 0b0000, MPAM Extension version 1.0 is implemented. If AArch64-ID_AA64PFR1_EL1.MPAM_frac == 0b0001, MPAM Extension version 1.1 is implemented. | |
| [39:36] | SEL2 | Secure EL2. Defined values are: 0b0001 Secure EL2 is implemented. | |
| [35:32] | SVE | Scalable Vector Extension. Defined values are: 0b0001 SVE architectural state and programmers' model are implemented. | |
| [31:28] | RAS | RAS Extension version. Defined values are: 0b0010 Armv8.4-RAS present. As 0b0001, and adds support for Armv8.4-DFE (If EL3 is implemented), additional ERXMISCm_EL1 System registers, additionalSystem registers ERXPFGCDN_EL1, ERXPFGCTL_EL1, and ERXPFGF_EL1, and the SCR_EL3.FIEN and HCR_EL2.FIEN trap controls, to support the optional RAS Common Fault Injection Model Extension. | |
| [27:24] | GIC | System register GIC CPU interface. Defined values are: 0b0000 When Port GICCDISABLE is High, GIC CPU interface is disabled. 0b0011 When Port GICCDISABLE is Low, GIC (version 4.1) CPU interface is enabled. | |

| Bits | Name | Description | Reset |
|---------|---------|---|-------|
| [23:20] | AdvSIMD | Advanced SIMD. Defined values are: 0b0001 Advanced SIMD is implemented, including support for half-precision floating-point arithmetic. | |
| [19:16] | FP | Floating-point. Defined values are: 0b0001 Floating-point, including support for half-precision floating-point arithmetic, is implemented. | |
| [15:12] | EL3 | EL3 Exception level handling. Defined values are: 0b0001 EL3 can be executed in AArch64 state only. | |
| [11:8] | EL2 | EL2 Exception level handling. Defined values are: 0b0001 EL2 can be executed in AArch64 state only. | |
| [7:4] | EL1 | EL1 Exception level handling. Defined values are: 0b0001 EL1 can be executed in AArch64 state only. | |
| [3:0] | ELO | ELO Exception level handling. Defined values are: 0b0001 ELO can be executed in AArch64 state only. | |

Access

MRS <Xt>, ID_AA64PFR0_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| ID_AA64PFR0_EL1 | 0b11 | 0b000 | 0b0000 | 0b0100 | 0b000 |

Accessibility

MRS <Xt>, ID_AA64PFR0_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64PFR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64PFR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64PFR0_EL1;

```

A.5.5 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1

Reserved for future expansion of information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-67: AArch64_id_aa64pfr1_el1 bit assignments

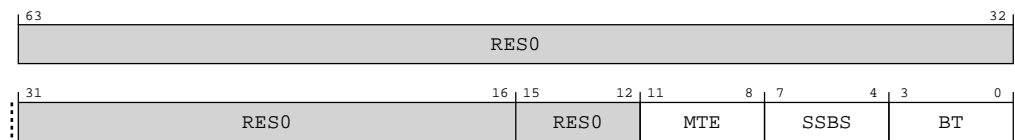


Table A-201: ID_AA64PFR1_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|---|--------|
| [63:12] | RES0 | Reserved | 0b0000 |
| [11:8] | MTE | Support for the Memory Tagging Extension. Defined values are: 0b0001 Memory Tagging Extension instructions accessible at EL0 are implemented. Instructions and System Registers defined by the extension not configurably accessible at EL0 are Unallocated and other System Register fields defined by the extension are RES0. This value is reported when the BROADCASTMTE input is LOW. 0b0011 Memory Tagging Extension is implemented with support for asymmetric Tag Check Fault handling. This value is reported when the BROADCASTMTE input is HIGH. | |
| [7:4] | SSBS | Speculative Store Bypassing controls in AArch64 state. Defined values are: 0b0010 AArch64 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypassing Safe, and the MSR and MRS instructions to directly read and write the PSTATE.SSBS field | |

| Bits | Name | Description | Reset |
|-------|------|---|-------|
| [3:0] | BT | Branch Target Identification mechanism support in AArch64 state. Defined values are: 0b0001 The Branch Target Identification mechanism is implemented. | |

Access

MRS <Xt>, ID_AA64PFR1_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| ID_AA64PFR1_EL1 | 0b11 | 0b000 | 0b0000 | 0b0100 | 0b001 |

Accessibility

MRS <Xt>, ID_AA64PFR1_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64PFR1_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64PFR1_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64PFR1_EL1;

```

A.5.6 ID_AA64ZFR0_EL1, SVE Feature ID register 0

Provides additional information about the implemented features of the AArch64 Scalable Vector Extension, when the AArch64-ID_AA64PFR0_EL1.SVE field is not zero.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations



Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

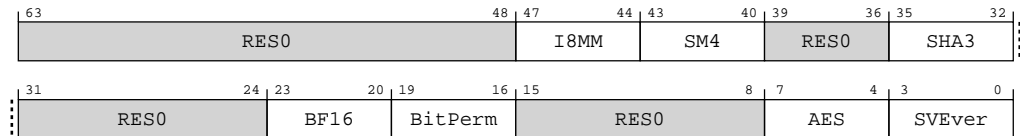
64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions**Figure A-68: AArch64_id_aa64zfr0_el1 bit assignments****Table A-203: ID_AA64ZFR0_EL1 bit descriptions**

| Bits | Name | Description | Reset |
|---------|---------|---|------------|
| [63:48] | RES0 | Reserved | 0x0 |
| [47:44] | I8MM | Indicates support for SVE Int8 matrix multiplication instructions. Defined values are: 0b0001 SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented. | |
| [43:40] | SM4 | Indicates support for SVE2 SM4 instructions. Defined values are: 0b0000 SVE2 SM4 instructions are not implemented. This value is reported when the Cryptographic Extension is not implemented or are disabled. 0b0001 SVE2 SM4E and SM4EKEY instructions are implemented. This value is reported when the Cryptographic Extension is implemented and enabled. | |
| [39:36] | RES0 | Reserved | 0b0000 |
| [35:32] | SHA3 | Indicates support for the SVE2 SHA3 instruction. Defined values are: 0b0000 SVE2 SHA3 instructions are not implemented. This value is reported when the Cryptographic Extension is not implemented or are disabled. 0b0001 SVE2 RAX1 instruction is implemented. This value is reported when the Cryptographic Extension is implemented and enabled. | |
| [31:24] | RES0 | Reserved | 0b00000000 |
| [23:20] | BF16 | Indicates support for SVE BFloat16 instructions. Defined values are: 0b0001 BFCVT, BFCVTNT, BFDOT, BFMLALB, BFMLALT, and BFMMMLA instructions are implemented. | |
| [19:16] | BitPerm | Indicates support for SVE2 bit permute instructions. Defined values are: 0b0001 SVE2 BDEP, BEXT and BGRP instructions are implemented. | |
| [15:8] | RES0 | Reserved | 0b00000000 |

| Bits | Name | Description | Reset |
|-------|--------|--|-------|
| [7:4] | AES | Indicates support for SVE2-AES instructions. Defined values are: 0b0000 SVE2-AES instructions are not implemented. This value is reported when the Cryptographic Extension is not implemented or are disabled. 0b0010 SVE2 AESE, AESD, AESMC, and AESIMC instructions are implemented plus SVE2 PMULLB and PMULLT instructions with 64-bit source. This value is reported when the Cryptographic Extension is implemented and enabled. | |
| [3:0] | SVEver | Scalable Vector Extension instruction set version. Defined values are: 0b0001 SVE and the non-optional SVE2 instructions are implemented. | |

Access

MRS <Xt>, ID_AA64ZFR0_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| ID_AA64ZFR0_EL1 | 0b11 | 0b000 | 0b0000 | 0b0100 | 0b100 |

Accessibility

MRS <Xt>, ID_AA64ZFR0_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (!IsZero(ID_AA64ZFR0_EL1) || boolean IMPLEMENTATION_DEFINED
        "ID_AA64ZFR0_EL1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ZFR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64ZFR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64ZFR0_EL1;

```

A.5.7 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0

Provides top-level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers, see Principles of the ID scheme for fields in ID registers.

Configurations

The external register ext-EDDFR gives information from this register.

Attributes**Width**

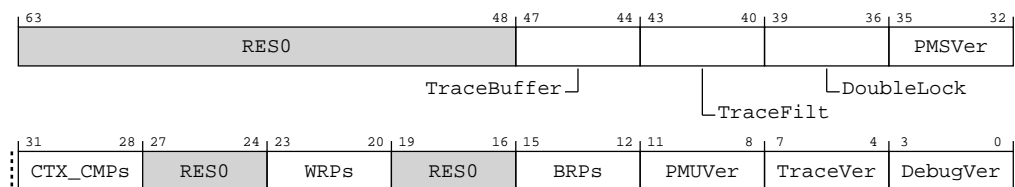
64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions**Figure A-69: AArch64_id_aa64dfr0_el1 bit assignments****Table A-205: ID_AA64DFR0_EL1 bit descriptions**

| Bits | Name | Description | Reset |
|---------|-------------|---|--------|
| [63:48] | RES0 | Reserved | 0x0 |
| [47:44] | TraceBuffer | Trace Buffer Extension version. Defined values are: 0b0001 Trace Buffer Extension implemented. | |
| [43:40] | TraceFilt | Armv8.4 Self-hosted Trace Extension version. Defined values are: 0b0001 Armv8.4 Self-hosted Trace Extension implemented. | |
| [39:36] | DoubleLock | OS Double Lock implemented. Defined values are: 0b1111 OS Double Lock not implemented. AArch64-OSDLR_EL1 is RAZ/WI. | |
| [35:32] | PMSVer | Statistical Profiling Extension version. | |
| [31:28] | CTX_CMPs | Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. 0b0001 Two context-aware breakpoints are included | |
| [27:24] | RES0 | Reserved | 0b0000 |
| [23:20] | WRPs | Number of watchpoints, minus 1. The value of 0b0000 is reserved. 0b0011 Four Watchpoints | |
| [19:16] | RES0 | Reserved | 0b0000 |
| [15:12] | BRPs | Number of breakpoints, minus 1. The value of 0b0000 is reserved. 0b0101 Six Breakpoints | |

| Bits | Name | Description | Reset |
|--------|----------|---|-------|
| [11:8] | PMUVer | Performance Monitors Extension version. Defined value is: 0b0110 Performance Monitors Extension implemented, PMUv3 for Armv8.5 | |
| [7:4] | TraceVer | Trace support. Indicates whether System register interface to a PE trace unit is implemented. Defined values are: 0b0001 PE trace unit System registers implemented. | |
| [3:0] | DebugVer | Debug architecture version. Indicates presence of Armv8 debug architecture. Defined values are: 0b1001 Armv8.4 debug architecture. | |

Access

MRS <Xt>, ID_AA64DFR0_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| ID_AA64DFR0_EL1 | 0b11 | 0b000 | 0b0000 | 0b0101 | 0b000 |

Accessibility

MRS <Xt>, ID_AA64DFR0_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64DFR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64DFR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64DFR0_EL1;

```

A.5.8 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1

Reserved for future expansion of top-level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

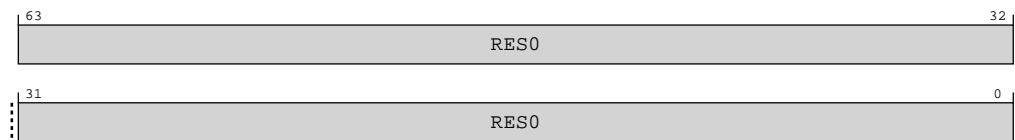
Functional group

Identification

Reset value

0x0

Bit descriptions

Figure A-70: AArch64_id_aa64dfr1_el1 bit assignments**Table A-207: ID_AA64DFR1_EL1 bit descriptions**

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, ID_AA64DFR1_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| ID_AA64DFR1_EL1 | 0b11 | 0b000 | 0b0000 | 0b0101 | 0b001 |

Accessibility

MRS <Xt>, ID_AA64DFR1_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64DFR1_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64DFR1_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64DFR1_EL1;

```

A.5.9 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0

Provides information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

0x0

Bit descriptions

Figure A-71: AArch64_id_aa64afr0_el1 bit assignments

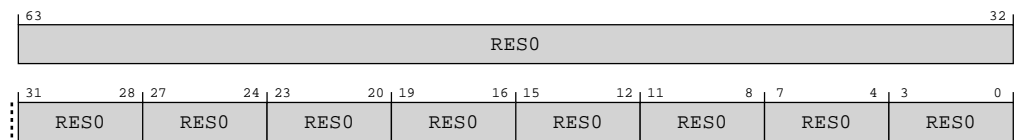


Table A-209: ID_AA64AFR0_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|--------|
| [63:0] | RES0 | Reserved | 0b0000 |

Access

MRS <Xt>, ID_AA64AFR0_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| ID_AA64AFR0_EL1 | 0b11 | 0b000 | 0b0000 | 0b0101 | 0b100 |

Accessibility

MRS <Xt>, ID_AA64AFR0_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then

```

```
if EL2Enabled() && HCR_EL2.TID3 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    return ID_AA64AFR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64AFR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64AFR0_EL1;
```

A.5.10 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1

Reserved for future expansion of information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

0x0

Bit descriptions

Figure A-72: AArch64_id_aa64afr1_el1 bit assignments

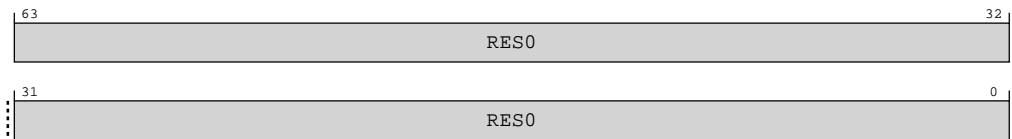


Table A-211: ID_AA64AFR1_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, ID_AA64AFR1_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| ID_AA64AFR1_EL1 | 0b11 | 0b000 | 0b0000 | 0b0101 | 0b101 |

Accessibility

MRS <Xt>, ID_AA64AFR1_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64AFR1_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64AFR1_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64AFR1_EL1;

```

A.5.11 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0

Provides information about the instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-73: AArch64_id_aa64isar0_el1 bit assignments

| | | | | | | | | | | | | | | | |
|------|------|--------|-------|------|------|-----|------|----|----|----|----|----|----|----|----|
| 63 | 60 | 59 | 56 | 55 | 52 | 51 | 48 | 47 | 44 | 43 | 40 | 39 | 36 | 35 | 32 |
| RNDR | TLB | TS | FHM | DP | SM4 | SM3 | SHA3 | | | | | | | | |
| 31 | 28 | 27 | 24 | 23 | 20 | 19 | 16 | 15 | 12 | 11 | 8 | 7 | 4 | 3 | 0 |
| RDM | RES0 | Atomic | CRC32 | SHA2 | SHA1 | AES | RES0 | | | | | | | | |

Table A-213: ID_AA64ISAR0_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|--|-------|
| [63:60] | RNDR | Indicates support for Random Number instructions in AArch64 state. Defined values are: 0b0000 No Random Number instructions are implemented. 0b0001 AArch64-RNDR and AArch64-RNDRS registers are implemented, if the core has the RNDR feature configured. | |
| [59:56] | TLB | Indicates support for Outer Shareable and TLB range maintenance instructions. Defined values are: 0b0010 Outer Shareable and TLB range maintenance instructions are implemented. | |
| [55:52] | TS | Indicates support for flag manipulation instructions. Defined values are: 0b0010 CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented. | |
| [51:48] | FHM | Indicates support for FMLAL and FMLS instructions. Defined values are: 0b0001 FMLAL and FMLS instructions are implemented. | |
| [47:44] | DP | Indicates support for Dot Product instructions in AArch64 state. Defined values are: 0b0001 UDOT and SDOT instructions implemented. | |
| [43:40] | SM4 | Indicates support for SM4 instructions in AArch64 state. Defined values are: 0b0000 When the Cryptographic Extension is not implemented or is disabled or the SM3/SM4 Cryptographic instructions are disabled, then SM4 instructions are not implemented. 0b0001 When the Cryptographic Extension is implemented and the SM3/SM4 Cryptographic instructions are enabled, then SM4 instructions SM4E and SM4EKEY are implemented. | |
| [39:36] | SM3 | Indicates support for SM3 instructions in AArch64 state. Defined values are: 0b0000 When the Cryptographic Extension is not implemented or is disabled or the SM3/SM4 Cryptographic instructions are disabled, then SM3 instructions are not implemented. 0b0001 When the Cryptographic Extension is implemented and the SM3/SM4 Cryptographic instructions are enabled, then SM3 instructions SM3SS1, SM3TT1A, SM3TT1B, SM3TT2A, SM3TT2B, SM3PARTW1, and SM3PARTW2 are implemented. | |
| [35:32] | SHA3 | Indicates support for SHA3 instructions in AArch64 state. Defined values are: 0b0000 When the Cryptographic Extension is not implemented or disabled then SHA3 instructions are not implemented. 0b0001 When the Cryptographic Extension is implemented and enabled then SHA3 instructions EOR3, RAX1, XAR, and BCAX are implemented. | |
| [31:28] | RDM | Indicates support for SQRDMLAH and SQRDMLSH instructions in AArch64 state. Defined values are: 0b0001 SQRDMLAH and SQRDMLSH instructions implemented. | |

| Bits | Name | Description | Reset |
|---------|--------|--|--------|
| [27:24] | RES0 | Reserved | 0b0000 |
| [23:20] | Atomic | Indicates support for Atomic instructions in AArch64 state. Defined values are: 0b0010 LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions implemented. | |
| [19:16] | CRC32 | CRC32 instructions implemented in AArch64 state. Defined values are: 0b0001 CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX instructions implemented. | |
| [15:12] | SHA2 | SHA2 instructions implemented in AArch64 state. Defined values are: 0b0000 When the Cryptographic Extension is not implemented or disabled then SHA2 instructions are not implemented. 0b0010 When the Cryptographic Extension is implemented and enabled then SHA256H, SHA256H2, SHA256SU0, SHA256SU1, SHA512H, SHA512H2, SHA512SU0, and SHA512SU1 instructions are implemented. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented | |
| [11:8] | SHA1 | SHA1 instructions implemented in AArch64 state. Defined values are: 0b0000 When the Cryptographic Extension is not implemented or disabled then SHA1 instructions are not implemented. 0b0001 When the Cryptographic Extension is implemented and enabled then SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions are implemented. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented | |
| [7:4] | AES | AES instructions implemented in AArch64 state. Defined values are: 0b0000 SVE2-AES instructions are not implemented. This value is reported when the Cryptographic Extension is not implemented or are disabled. 0b0010 SVE2 AESE, AESD, AESMC, and AESIMC instructions are implemented plus SVE2 PMULLB and PMULLT instructions with 64-bit source. This value is reported when the Cryptographic Extension is implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented | |
| [3:0] | RES0 | Reserved | 0b0000 |

Access

MRS <Xt>, ID_AA64ISAR0_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|------------------|------|-------|--------|--------|-------|
| ID_AA64ISAR0_EL1 | 0b11 | 0b000 | 0b0000 | 0b0110 | 0b000 |

Accessibility

MRS <Xt>, ID_AA64ISAR0_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ISAR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64ISAR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64ISAR0_EL1;

```

A.5.12 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

If ID_AA64ISAR1_EL1.{API, APA} == {0000, 0000}, then:

- The AArch64-TCR_EL1.{TBID,TBID0}, AArch64-TCR_EL2.{TBID0,TBID1}, AArch64-TCR_EL2.TBID and AArch64-TCR_EL3.TBID bits are RES0.
- AArch64-APIAKeyHi_EL1, AArch64-APIAKeyLo_EL1, AArch64-APIBKeyHi_EL1, AArch64-APIBKeyLo_EL1, AArch64-APDAKeyHi_EL1, AArch64-APDAKeyLo_EL1, AArch64-APDBKeyHi_EL1, AArch64-APDBKeyLo_EL1 are not allocated.
- 'SCTLR_EL'.EnIA, 'SCTLR_EL'.EnIB, 'SCTLR_EL'.EnDA, 'SCTLR_EL'.EnDB are all RES0.

If ID_AA64ISAR1_EL1.{GPI, GPA, API, APA} == {0000, 0000, 0000, 0000}, then:

- AArch64-HCR_EL2.APK and AArch64-HCR_EL2.API are RES0.
- AArch64-SCR_EL3.APK and AArch64-SCR_EL3.API are RES0.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-74: AArch64_id_aa64isar1_el1 bit assignments

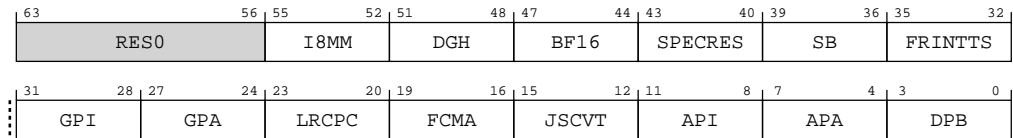


Table A-215: ID_AA64ISAR1_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|---------|---|------------|
| [63:56] | RES0 | Reserved | 0b00000000 |
| [55:52] | I8MM | Indicates support for Advanced SIMD and Floating-point Int8 matrix multiplication instructions in AArch64 state. Defined values of this field are: 0b0001 SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented. | |
| [51:48] | DGH | Indicates support for the Data Gathering Hint instruction. Defined values are: 0b0001 Data Gathering Hint is implemented. | |
| [47:44] | BF16 | Indicates support for Advanced SIMD and Floating-point BFloat16 instructions in AArch64 state. Defined values are: 0b0001 BFDOT, BFMLAL, BFMLAL2, BFMMMLA, BFCVT, and BFCVT2 instructions are implemented. | |
| [43:40] | SPECRES | Indicates support for prediction invalidation instructions in AArch64 state. Defined values are: 0b0001 CFP RCTX, DVP RCTX, and CPP RCTX instructions are implemented. | |
| [39:36] | SB | Indicates support for SB instruction in AArch64 state. Defined values are: 0b0001 SB instruction is implemented. | |
| [35:32] | FRINTTS | Indicates support for the FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented. Defined values are: 0b0001 FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented. | |
| [31:28] | GPI | Indicates support for an IMPLEMENTATION DEFINED algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: 0b0000 Generic Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented. | |
| [27:24] | GPA | Indicates whether QARMA or Architected algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: 0b0001 Generic Authentication using the QARMA algorithm is implemented. This includes the PACGA instruction. | |

| Bits | Name | Description | Reset |
|---------|-------|---|-------|
| [23:20] | LRCPC | Indicates support for weaker release consistency, RCpc, based model. Defined values are: 0b0010 The LDAPR*, LDAPUR*, and STLUR* instructions are implemented. | |
| [19:16] | FCMA | Indicates support for complex number addition and multiplication, where numbers are stored in vectors. Defined values are: 0b0001 The FCMLA and FCADD instructions are implemented. | |
| [15:12] | JSCVT | Indicates support for JavaScript conversion from double precision floating point values to integers in AArch64 state. Defined values are: 0b0001 The FJCVTZS instruction is implemented. | |
| [11:8] | API | Indicates whether an IMPLEMENTATION DEFINED algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are: 0b0000 Address Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented. | |
| [7:4] | APA | Indicates whether QARMA or Architected algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are: 0b0101 Address Authentication using the QARMA algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE. | |
| [3:0] | DPB | Data Persistence writeback. Indicates support for the rDC CVAP and rDC CVADP instructions in AArch64 state. Defined values are: 0b0010 rDC CVAP and rDC CVADP supported. | |

Access

MRS <Xt>, ID_AA64ISAR1_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|------------------|------|-------|--------|--------|-------|
| ID_AA64ISAR1_EL1 | 0b11 | 0b000 | 0b0000 | 0b0110 | 0b001 |

Accessibility

MRS <Xt>, ID_AA64ISAR1_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ISAR1_EL1;
elseif PSTATE.EL == EL2 then

```

```

return ID_AA64ISAR1_EL1;
elseif PSTATE.EL == EL3 then
return ID_AA64ISAR1_EL1;

```

A.5.13 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-75: AArch64_id_aa64mmfr0_el1 bit assignments

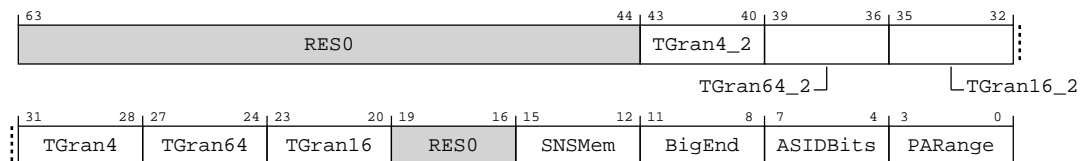


Table A-217: ID_AA64MMFR0_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|-----------|--|-------|
| [63:44] | RES0 | Reserved | 0x0 |
| [43:40] | TGran4_2 | Indicates support for 4KB memory granule size for stage 2. Defined values are: 0b0010 4KB granule supported at stage 2. | |
| [39:36] | TGran64_2 | Indicates support for 64KB memory granule size for stage 2. Defined values are: 0b0010 64KB granule supported at stage 2. | |

| Bits | Name | Description | Reset |
|---------|-----------|--|--------|
| [35:32] | TGran16_2 | Indicates support for 16KB memory granule size for stage 2. Defined values are: 0b0010 16KB granule supported at stage 2 | |
| [31:28] | TGran4 | Indicates support for 4KB memory translation granule size. Defined values are: 0b0000 4KB granule supported. | |
| [27:24] | TGran64 | Indicates support for 64KB memory translation granule size. Defined values are: 0b0000 64KB granule supported. | |
| [23:20] | TGran16 | Indicates support for 16KB memory translation granule size. Defined values are: 0b0001 16KB granule supported. | |
| [19:16] | RES0 | Reserved | 0b0000 |
| [15:12] | SNSMem | Indicates support for a distinction between Secure and Non-secure Memory. Defined values are: 0b0001 Does support a distinction between Secure and Non-secure Memory. | |
| [11:8] | BigEnd | Indicates support for mixed-endian configuration. Defined values are: 0b0001 Mixed-endian support. The SCTLR_ELx.EE and SCTLR_EL1.EOE bits can be configured. | |
| [7:4] | ASIDBits | Number of ASID bits. Defined values are: 0b0010 16 bits. | |
| [3:0] | PARange | Physical Address range supported. Defined values are: 0b0101 48 bits, 256TB. | |

Access

MRS <Xt>, ID_AA64MMFRO_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|------------------|------|-------|--------|--------|-------|
| ID_AA64MMFRO_EL1 | 0b11 | 0b000 | 0b0000 | 0b0111 | 0b000 |

Accessibility

MRS <Xt>, ID_AA64MMFRO_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFRO_EL1;
elseif PSTATE.EL == EL2 then

```

```

return ID_AA64MMFR0_EL1;
elseif PSTATE.EL == EL3 then
return ID_AA64MMFR0_EL1;

```

A.5.14 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-76: AArch64_id_aa64mmfr1_el1 bit assignments

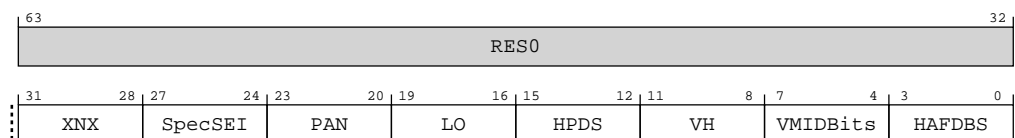


Table A-219: ID_AA64MMFR1_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|---------|---|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:28] | XNX | Indicates support for execute-never control distinction by Exception level at stage 2. Defined values are: 0b0001 Distinction between EL0 and EL1 execute-never control at stage 2 supported. | |
| [27:24] | SpecSEI | Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches. The defined values of this field are: 0b0000 The PE never generates an SError interrupt due to an External abort on a speculative read. | |

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [23:20] | PAN | Privileged Access Never. Indicates support for the PAN bit in PSTATE, AArch64-SPSR_EL1, AArch64-SPSR_EL2, AArch64-SPSR_EL3, and AArch64-DSPSR_EL0. Defined values are: 0b0011 PAN supported, AT S1E1RP and AT S1E1WP instructions supported, and AArch64-SCTLR_EL1.EPAN and AArch64-SCTLR_EL2.EPAN bits supported | |
| [19:16] | LO | LORegions. Indicates support for LORegions. Defined values are: 0b0001 LORegions supported. | |
| [15:12] | HPDS | Hierarchical Permission Disables. Indicates support for disabling hierarchical controls in translation tables. Defined values are: 0b0010 Disabling of hierarchical controls supported with the TCR_EL1.{HPD1, HPD0}, TCR_EL2.HPD or TCR_EL2.{HPD1, HPD0}, and TCR_EL3.HPD bits and adds possible hardware allocation of bits[62:59] of the translation table descriptors from the final lookup level for IMPLEMENTATION DEFINED use. | |
| [11:8] | VH | Virtualization Host Extensions. Defined values are: 0b0001 Virtualization Host Extensions supported. | |
| [7:4] | VMIDBits | Number of VMID bits. Defined values are: 0b0010 16 bits | |
| [3:0] | HAFDBS | Hardware updates to Access flag and Dirty state in translation tables. Defined values are: 0b0010 Hardware update of both the Access flag and dirty state is supported. | |

Access

MRS <Xt>, ID_AA64MMFR1_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|------------------|------|-------|--------|--------|-------|
| ID_AA64MMFR1_EL1 | 0b11 | 0b000 | 0b0000 | 0b0111 | 0b001 |

Accessibility

MRS <Xt>, ID_AA64MMFR1_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFR1_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64MMFR1_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64MMFR1_EL1;

```

A.5.15 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations



Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-77: AArch64_id_aa64mmfr2_el1 bit assignments

| | | | | | | | | | | | | | | | |
|------|----|-----|----|-------|----|------|----|------|----|------|----|-----|----|-----|----|
| 63 | 60 | 59 | 56 | 55 | 52 | 51 | 48 | 47 | 44 | 43 | 40 | 39 | 36 | 35 | 32 |
| EOPD | | EVT | | BBM | | TTL | | RES0 | | FWB | | IDS | | AT | |
| 31 | 28 | 27 | 24 | 23 | 20 | 19 | 16 | 15 | 12 | 11 | 8 | 7 | 4 | 3 | 0 |
| ST | | NV | | CCIDX | | RES0 | | IESB | | RES0 | | UAO | | CnP | |

Table A-221: ID_AA64MMFR2_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|---|-------|
| [63:60] | EOPD | Indicates support for the EOPD mechanism. Defined values are: 0b0001 EOPDx mechanism is implemented. | |
| [59:56] | EVT | Enhanced Virtualization Traps. If EL2 is implemented, indicates support for the AArch64-HCR_EL2.{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps. Defined values are: 0b0010 AArch64-HCR_EL2.{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps are supported. | |

| Bits | Name | Description | Reset |
|---------|-------|--|--------|
| [55:52] | BBM | Allows identification of the requirements of the hardware to have break-before-make sequences when changing block size for a translation. 0b0010 Level 2 support for changing block size is supported. | |
| [51:48] | TTL | Indicates support for TTL field in address operations. Defined values are: 0b0001 TLB maintenance instructions by address have bits[47:44] holding the TTL field. | |
| [47:44] | RES0 | Reserved | 0b0000 |
| [43:40] | FWB | Indicates support for AArch64-HCR_EL2.FWB. Defined values are: 0b0001 AArch64-HCR_EL2.FWB is supported. | |
| [39:36] | IDS | Indicates the value of ESR_ELx.EC that reports an exception generated by a read access to the feature ID space. Defined values are: 0b0001 All exceptions generated by an AArch64 read access to the feature ID space are reported by ESR_ELx.EC == 0x18. | |
| [35:32] | AT | Identifies support for unaligned single-copy atomicity and atomic functions. Defined values are: 0b0001 Unaligned single-copy atomicity and atomic functions with a 16-byte address range aligned to 16-bytes are supported. | |
| [31:28] | ST | Identifies support for small translation tables. Defined values are: 0b0001 The maximum value of the TCR_ELx.{TOSZ,T1SZ} and VTCR_EL2.TOSZ fields is 48 for 4KB and 16KB granules, and 47 for 64KB granules. | |
| [27:24] | NV | Nested Virtualization. If EL2 is implemented, indicates support for the use of nested virtualization. Defined values are: 0b0010 The AArch64-VNCR_EL2 register and the HCR_EL2.{AT, NV, NV1, NV2} bits are implemented. | |
| [23:20] | CCIDX | Support for the use of revised AArch64-CCSIDR_EL1 register format. Defined values are: 0b0001 64-bit format implemented for all levels of the CCSIDR_EL1. | |
| [19:16] | RES0 | Reserved | 0b0000 |
| [15:12] | IESB | Indicates support for the IESB bit in the SCTLR_ELx registers. Defined values are: 0b0001 IESB bit in the SCTLR_ELx registers is supported. | |
| [11:8] | RES0 | Reserved | 0b0000 |
| [7:4] | UAO | User Access Override. Defined values are: 0b0001 UAO supported. | |
| [3:0] | CnP | Indicates support for Common not Private translations. Defined values are: 0b0001 Common not Private translations supported. | |

Access

MRS <Xt>, ID_AA64MMFR2_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|------------------|------|-------|--------|--------|-------|
| ID_AA64MMFR2_EL1 | 0b11 | 0b000 | 0b0000 | 0b0111 | 0b010 |

Accessibility

MRS <Xt>, ID_AA64MMFR2_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (!IsZero(ID_AA64MMFR2_EL1) || boolean IMPLEMENTATION_DEFINED
        "ID_AA64MMFR2 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFR2_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64MMFR2_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64MMFR2_EL1;

```

A.5.16 CLIDR_EL1, Cache Level ID Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-78: AArch64_clidr_el1 bit assignments

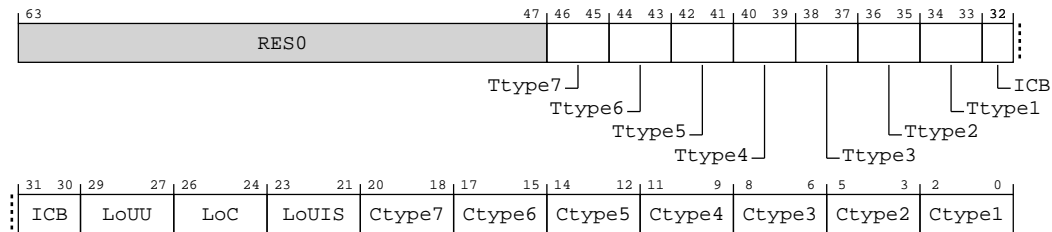


Table A-223: CLIDR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|--------|---|-------|
| [63:47] | RES0 | Reserved | 0x0 |
| [46:45] | Ttype7 | Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. 0b00 No Tag Cache. | |
| [44:43] | Ttype6 | Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. 0b00 No Tag Cache. | |
| [42:41] | Ttype5 | Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. 0b00 No Tag Cache. | |
| [40:39] | Ttype4 | Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. 0b00 No Tag Cache. | |
| [38:37] | Ttype3 | Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. 0b00 When no L3 present, no tag cache. 0b10 When L3 present, Unified Allocation Tag and Data cache at L3 | |
| [36:35] | Ttype2 | Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. 0b10 Unified Allocation Tag and Data cache at L1 | |

| Bits | Name | Description | Reset |
|---------|--------|---|-------|
| [34:33] | Ttype1 | Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. 0b10 Unified Allocation Tag and Data cache at L1 | |
| [32:30] | ICB | Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions. The possible values are: 0b000 Not disclosed by this mechanism. 0b001 L1 cache is the highest Inner Cacheable level. 0b010 L2 cache is the highest Inner Cacheable level. 0b011 L3 cache is the highest Inner Cacheable level. 0b100 L4 cache is the highest Inner Cacheable level. 0b101 L5 cache is the highest Inner Cacheable level. 0b110 L6 cache is the highest Inner Cacheable level. 0b111 L7 cache is the highest Inner Cacheable level. | |
| [29:27] | LoUU | Level of Unification Uniprocessor for the cache hierarchy. 0b000 Level of Unification Uniprocessor is before the L1 data cache. | |
| [26:24] | LoC | Level of Coherence for the cache hierarchy. 0b010 When no L3 present, Level 2 0b011 When L3 present, Level 3 | |
| [23:21] | LoUIS | Level of Unification Inner Shareable for the cache hierarchy. 0b000 No cache level needs cleaning to Point of Unification | |
| [20:18] | Ctype7 | Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: 0b000 No cache. | |

| Bits | Name | Description | Reset |
|---------|--------|---|-------|
| [17:15] | Ctype6 | Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: 0b000 No cache. | |
| [14:12] | Ctype5 | Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: 0b000 No cache. | |
| [11:9] | Ctype4 | Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: 0b000 No cache. | |
| [8:6] | Ctype3 | Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: 0b000 No L3. 0b100 Unified instruction and data caches at L3 | |
| [5:3] | Ctype2 | Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: 0b100 Unified instruction and data caches at L2 | |
| [2:0] | Ctype1 | Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: 0b011 Separate instruction and data caches at L1 | |

Access

MRS <Xt>, CLIDR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| CLIDR_EL1 | 0b11 | 0b001 | 0b0000 | 0b0000 | 0b001 |

Accessibility

MRS <Xt>, CLIDR_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);

```

```
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.TID4 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.CLIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CLIDR_EL1;
elseif PSTATE.EL == EL2 then
    return CLIDR_EL1;
elseif PSTATE.EL == EL3 then
    return CLIDR_EL1;
```

A.5.17 GMID_EL1, Multiple tag transfer ID register

Indicates the block size that is accessed by the LDGM and STGM System instructions.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-79: AArch64_gmid_el1 bit assignments

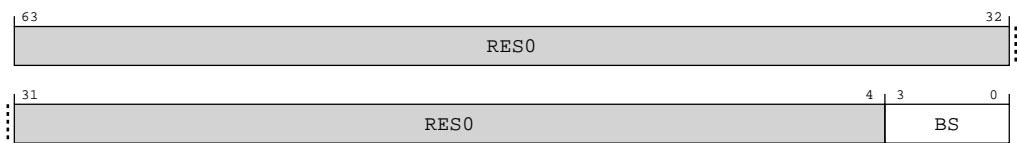


Table A-225: GMID_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|--|-------|
| [63:4] | RES0 | Reserved | 0x0 |
| [3:0] | BS | Log ₂ of the block size in words. The minimum supported size is 16B (value == 2) and the maximum is 256B (value == 6). 0b0100 Log ₂ of the block size is 4 | |

Access

MRS <Xt>, GMID_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| GMID_EL1 | 0b11 | 0b001 | 0b0000 | 0b0000 | 0b100 |

Accessibility

MRS <Xt>, GMID_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID5 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return GMID_EL1;
elseif PSTATE.EL == EL2 then
    return GMID_EL1;
elseif PSTATE.EL == EL3 then
    return GMID_EL1;

```

A.5.18 CTR_EL0, Cache Type Register

Provides information about the architecture of the caches.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

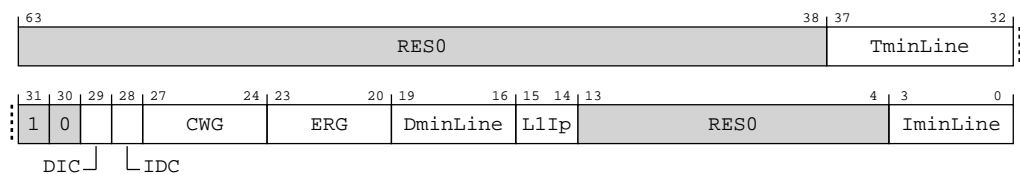
Figure A-80: AArch64_ctr_el0 bit assignments

Table A-227: CTR_EL0 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [63:38] | RES0 | Reserved | 0x0 |
| [37:32] | TminLine | <p>Tag minimum Line. \log_2 of the number of words covered by Allocation Tags in the smallest cache line of all caches which can contain Allocation tags that are controlled by the PE.</p> <p>Note:</p> <ul style="list-style-type: none"> For an implementation with cache lines containing 64 bytes of data and 4 Allocation Tags, this will be $\log_2(64/4) = 4$. For an implementation with Allocations Tags in separate cache lines of 128 Allocation Tags per line, this will be $\log_2(128*16/4) = 9$. <p>0b000100</p> <p>\log_2 of number of words (64/4=16) covered by Allocation Tags in the smallest cache line of all caches</p> | |
| [31] | RES1 | Reserved | 0b1 |
| [30] | RES0 | Reserved | 0b0 |
| [29] | DIC | <p>Instruction cache invalidation requirements for data to instruction coherence.</p> <p>0b0</p> <p>When COHERENT_ICACHE not enabled, Instruction cache invalidation to the point of unification is required for instruction to data coherence.</p> <p>0b1</p> <p>When COHERENT_ICACHE enabled, Instruction cache cleaning to the point of unification is not required for instruction to data coherence.</p> | |
| [28] | IDC | <p>Data cache clean requirements for instruction to data coherence. The meaning of this bit is:</p> <p>0b1</p> <p>Data cache clean to the Point of Unification is not required for instruction to data coherence.</p> | |
| [27:24] | CWG | <p>Cache writeback 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.</p> <p>0b0100</p> <p>64 bytes.</p> | |
| [23:20] | ERG | <p>Exclusives reservation granule, and, if TME is implemented, transactional reservation granule. \log_2 of the number of words of the maximum size of the reservation granule for the Load-Exclusive and Store-Exclusive instructions, and, if TME is implemented, for detecting transactional conflicts.</p> <p>0b0100</p> <p>64 bytes.</p> | |
| [19:16] | DminLine | <p>\log_2 of the number of words in the smallest cache line of all the data caches and unified caches that are controlled by the PE.</p> <p>0b0100</p> <p>64 bytes.</p> | |
| [15:14] | L1Ip | <p>Level 1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache. Possible values of this field are:</p> <p>0b11</p> <p>Physical Index, Physical Tag (PIPT)</p> | |
| [13:4] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|----------|---|-------|
| [3:0] | lminLine | Log ₂ of the number of words in the smallest cache line of all the instruction caches that are controlled by the PE. 0b0100 64 bytes. | |

Access

MRS <Xt>, CTR_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| CTR_ELO | 0b11 | 0b011 | 0b0000 | 0b0000 | 0b001 |

Accessibility

MRS <Xt>, CTR_ELO

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.UCT == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TID2 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
            HFGTR_EL2.CTR_ELO == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCT == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            return CTR_ELO;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TID2 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.CTR_ELO == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            return CTR_ELO;
    elsif PSTATE.EL == EL2 then
        return CTR_ELO;
    elsif PSTATE.EL == EL3 then
        return CTR_ELO;

```

A.5.19 DCZID_ELO, Data Cache Zero ID register

Indicates the block size that is written with byte values of 0 by the rDC ZVA (Data Cache Zero by Address) System instruction.

If Armv8.5-MemTag is implemented, this register also indicates the granularity at which the rDC GVA and rDC GZVA instructions write.

Configurations

This register is available in all configurations.

Attributes

Width

64

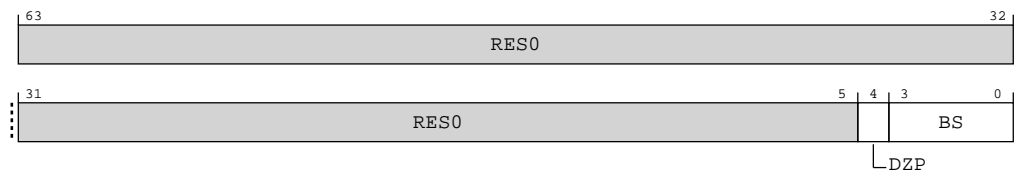
Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

Figure A-81: AArch64_dczip_el0 bit assignments**Table A-229: DCZID_EL0 bit descriptions**

| Bits | Name | Description | Reset |
|--------|------|--|-------|
| [63:5] | RES0 | Reserved | 0x0 |
| [4] | DZP | Data Zero Prohibited. This field indicates whether use of rDC ZVA instructions is permitted or prohibited. If Armv8.5-MemTag is implemented, this field also indicates whether use of the rDC GVA and rDC GZVA instructions are permitted or prohibited. 0b0 Instructions are permitted. | |
| [3:0] | BS | Log ₂ of the block size in words. The maximum size supported is 2KB (value == 9). 0b0100 Log ₂ of the block size is 4 | |

Access

MRS <Xt>, DCZID_EL0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| DCZID_EL0 | 0b11 | 0b011 | 0b0000 | 0b0000 | 0b111 |

Accessibility

MRS <Xt>, DCZID_EL0

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
        HFGRTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return DCZID_EL0;

```

```
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return DCZID_EL0;
elseif PSTATE.EL == EL2 then
    return DCZID_EL0;
elseif PSTATE.EL == EL3 then
    return DCZID_EL0;
```

A.5.20 MPAMIDR_EL1, MPAM ID Register (EL1)

Indicates the presence and maximum PARTID and PMG values supported in the implementation. It also indicates whether the implementation supports MPAM virtualization.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions

MPAMIDR_EL1 indicates the MPAM implementation parameters of the PE.

Figure A-82: AArch64_mpamidr_el1 bit assignments

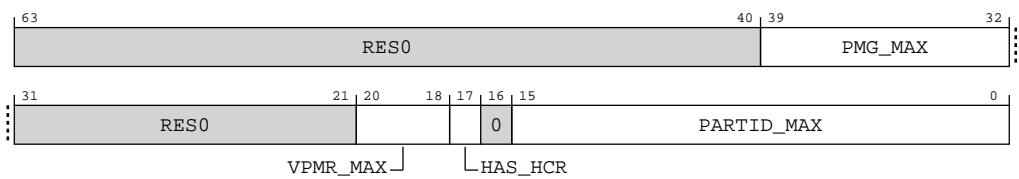


Table A-231: MPAMIDR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|---------|--|-------|
| [63:40] | RES0 | Reserved | 0x0 |
| [39:32] | PMG_MAX | The largest value of PMG that the implementation can generate. The PMG_I and PMG_D fields of every MPAMn_ELx must implement at least enough bits to represent PMG_MAX. 0b00000001 Max PMG field is 1 (1-bit) | |
| [31:21] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|---------|------------|---|-------|
| [20:18] | VPMR_MAX | If HAS_HCR == 0, VPMR_MAX must be 0b000. Otherwise, it indicates the maximum register index n for the MPAMVPM<n>_EL2 registers. 0b111 8 MPAMVPMn_EL2 registers are implemented | |
| [17] | HAS_HCR | HAS_HCR indicates that the PE implementation supports MPAM virtualization, including AArch64-MPAMHCR_EL2, AArch64-MPAMVPMV_EL2 and MPAMVPM<n>_EL2 with n in the range 0 to VPMR_MAX. Must be 0 if EL2 is not implemented in either security state. 0b1 MPAM virtualization is supported. | |
| [16] | RES0 | Reserved | 0b0 |
| [15:0] | PARTID_MAX | The largest value of PARTID that the implementation can generate. The PARTID_I and PARTID_D fields of every MPAMn_ELx must implement at least enough bits to represent PARTID_MAX. 0b0000000111111111 Max PARTID field is 511 | |

Access

MRS <Xt>, MPAMIDR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| MPAMIDR_EL1 | 0b11 | 0b000 | 0b1010 | 0b0100 | 0b100 |

Accessibility

MRS <Xt>, MPAMIDR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elseif EL2Enabled() && MPAMHCR_EL2.TRAP_MPAMIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return MPAMIDR_EL1;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMIDR_EL1;
elseif PSTATE.EL == EL3 then
    return MPAMIDR_EL1;

```

A.5.21 IMP_CPUCFR_EL1, CPU Configuration Register

This register provides configuration information for the core.

Configurations

This register is available in all configurations.

Attributes**Width**

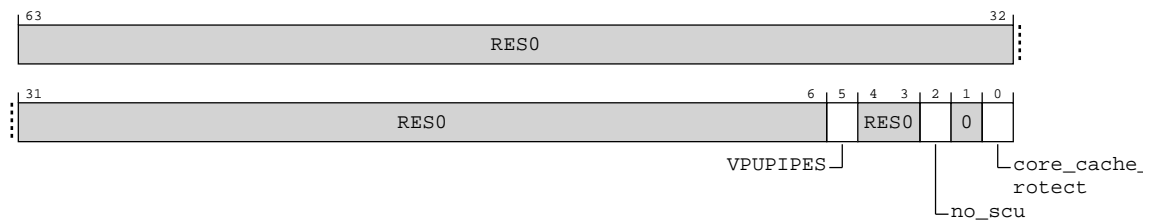
64

Functional group

Identification

Reset value

See individual bit resets

Bit descriptions**Figure A-83: AArch64_imp_cpucfr_el1 bit assignments****Table A-233: IMP_CPUCFR_EL1 bit descriptions**

| Bits | Name | Description | Reset |
|--------|--------------------|---|-------|
| [63:6] | RES0 | Reserved | 0x0 |
| [5] | VPUPIPIES | Indicates the number of Vector Processing Unit (VPU) pipes. Possible values of this bit are: 0b1 4 x 128-bit | |
| [4:3] | RES0 | Reserved | 0b00 |
| [2] | no_scu | Indicates whether the SCU is present or not. Possible values of this bit are: 0b0 The SCU is present. 0b1 The SCU is not present. | |
| [1] | RES0 | Reserved | 0b0 |
| [0] | core_cache_protect | Indicates whether ECC is present or not. Possible values of this field are: 0b0 ECC is not present. 0b1 ECC is present. | |

Access

MRS <Xt>, S3_0_C15_C0_0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| S3_0_C15_C0_0 | 0b11 | 0b000 | 0b1111 | 0b0000 | 0b000 |

Accessibility

MRS <Xt>, S3_0_C15_C0_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUCFR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUCFR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUCFR_EL1;
```

A.6 AArch64 Performance monitors register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** performance monitors registers in the core. For more information about a register, you can click the register name in the table.

Table A-235: AArch64 Performance monitors register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-----------------------------|-----|-----|-----|-----|-----|---------------------------|--------|---|
| PMMIR_EL1 | 3 | C9 | 0 | C14 | 6 | See individual bit resets | 64-bit | Performance Monitors Machine Identification Register |
| PMCR_ELO | 3 | C9 | 3 | C12 | 0 | See individual bit resets | 64-bit | Performance Monitors Control Register |
| PMCEID0_ELO | 3 | C9 | 3 | C12 | 6 | See individual bit resets | 64-bit | Performance Monitors Common Event Identification register 0 |
| PMCEID1_ELO | 3 | C9 | 3 | C12 | 7 | See individual bit resets | 64-bit | Performance Monitors Common Event Identification register 1 |

A.6.1 PMMIR_EL1, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation to software.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-84: AArch64_pmmir_el1 bit assignments

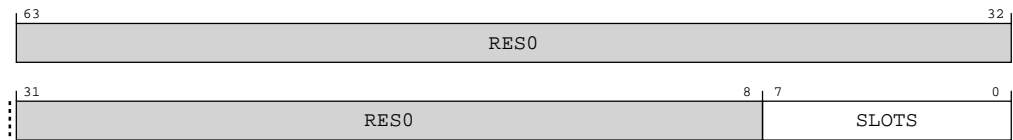


Table A-236: PMMIR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|--|-------|
| [63:8] | RES0 | Reserved | 0x0 |
| [7:0] | SLOTS | Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero. 0b00001000 The largest value by which the STALL_SLOT PMU event may increment in one cycle is 8. | |

Access

MRS <Xt>, PMMIR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| PMMIR_EL1 | 0b11 | 0b000 | 0b1001 | 0b1110 | 0b110 |

Accessibility

MRS <Xt>, PMMIR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMMIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMMIR_EL1;
elseif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMMIR_EL1;
elseif PSTATE.EL == EL3 then
    return PMMIR_EL1;

```

A.6.2 PMCR_EL0, Performance Monitors Control Register

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-85: AArch64_pmcr_el0 bit assignments

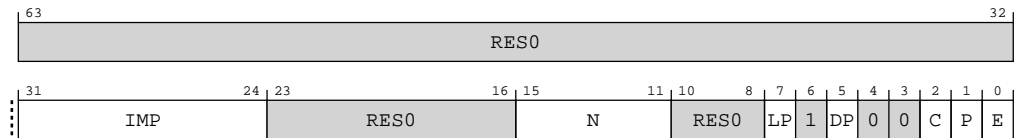


Table A-238: PMCR_EL0 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|---|------------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:24] | IMP | Implementer code: 0b00000000 No ID information is present in PMCR/PMCR_EL0. Software must use the MIDR_EL1 to identify the PE. | |
| [23:16] | RES0 | Reserved | 0b00000000 |
| [15:11] | N | Number of event counters: 0b00110 6 PMU counters implemented | |
| [10:8] | RES0 | Reserved | 0b000 |

| Bits | Name | Description | Reset |
|-------|------|--|-------|
| [7] | LP | <p>Long event counter enable. Determines when unsigned overflow is recorded by a counter overflow bit.</p> <p>0b0</p> <p>Event counter overflow on increment that causes unsigned overflow of AArch64-PMEVCNTR<n>_ELO[31:0].</p> <p>0b1</p> <p>Event counter overflow on increment that causes unsigned overflow of AArch64-PMEVCNTR<n>_ELO[63:0].</p> <p>If EL2 is implemented and AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN is less than PMCR_ELO.N, this bit does not affect the operation of event counters in the range [AArch32-HDCR.HPMN..(PMCR_ELO.N-1)] or [AArch64-MDCR_EL2.HPMN..(PMCR_ELO.N-1)].</p> <p>Note:</p> <p>The effect of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state. For more information, see the description of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN.</p> | |
| [6] | RES1 | Reserved | 0b1 |
| [5] | DP | <p>Disable cycle counter when event counting is prohibited.</p> <p>0b0</p> <p>Cycle counting by AArch64-PMCCNTR_ELO is not affected by this bit.</p> <p>0b1</p> <p>When event counting for counters in the range [0..(AArch64-MDCR_EL2.HPMN-1)] is prohibited, cycle counting by AArch64-PMCCNTR_ELO is disabled.</p> | |
| [4:3] | RES0 | Reserved | 0b0 |
| [2] | C | <p>Cycle counter reset. The effects of writing to this bit are:</p> <p>0b0</p> <p>No action.</p> <p>0b1</p> <p>Reset AArch64-PMCCNTR_ELO to zero.</p> <p>This bit is always RAZ.</p> <p>Note:</p> <p>Resetting AArch64-PMCCNTR_ELO does not change the cycle counter overflow bit. The value of PMCR_ELO.LC is ignored, and bits [63:0] of all affected event counters are reset.</p> | |

| Bits | Name | Description | Reset |
|------|------|---|-------|
| [1] | P | <p>Event counter reset. The effects of writing to this bit are:</p> <p>0b0</p> <p>No action.</p> <p>0b1</p> <p>Reset all event counters accessible in the current Exception level, not including AArch64-PMCCNTR_ELO, to zero.</p> <p>This bit is always RAZ.</p> <p>In EL0 and EL1:</p> <ul style="list-style-type: none"> If EL2 is implemented and enabled in the current Security state, and AArch64-MDCR_EL2.HPMN is less than PMCR_ELO.N, a write of 1 to this bit does not reset event counters in the range [AArch64-MDCR_EL2.HPMN..(PMCR_ELO.N-1)]. If EL2 is not implemented, EL2 is disabled in the current Security state, or AArch64-MDCR_EL2.HPMN equals PMCR_ELO.N, a write of 1 to this bit resets all the event counters. <p>In EL2 and EL3, a write of 1 to this bit resets all the event counters.</p> <p>Note: Resetting the event counters does not change the event counter overflow bits.</p> <p>If Armv8.5-PMU is implemented, the values of AArch64-MDCR_EL2.HLP and PMCR_ELO.LP are ignored, and bits [63:0] of all affected event counters are reset.</p> | |
| [0] | E | <p>Enable.</p> <p>0b0</p> <p>All event counters in the range [0..(PMN-1)] and AArch64-PMCCNTR_ELO, are disabled.</p> <p>0b1</p> <p>All event counters in the range [0..(PMN-1)] and AArch64-PMCCNTR_ELO, are enabled by AArch64-PMCNTENSET_ELO.</p> <p>If EL2 is implemented, then:</p> <ul style="list-style-type: none"> If EL2 is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. If PMN is less than PMCR_ELO.N, this bit does not affect the operation of event counters in the range [PMN..(PMCR_ELO.N-1)]. <p>If EL2 is not implemented, PMN is PMCR_ELO.N.</p> <p>Note: The effect of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state. For more information, see the description of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN.</p> | |

Access

MRS <Xt>, PMCR_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| PMCR_ELO | 0b11 | 0b011 | 0b1001 | 0b1100 | 0b000 |

MSR PMCR_ELO, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| PMCR_ELO | 0b11 | 0b011 | 0b1001 | 0b1100 | 0b000 |

Accessibility

MRS <Xt>, PMCR_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCR_ELO;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCR_ELO;
    elsif PSTATE.EL == EL2 then
        if MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCR_ELO;
    elsif PSTATE.EL == EL3 then
        return PMCR_ELO;

```

MSR PMCR_ELO, <Xt>

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
HDFGWTR_EL2.PMCR_ELO == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCR_ELO = X[t];
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMCR_ELO == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);

```

```

    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t];
    elsif PSTATE.EL == EL2 then
        if MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCR_EL0 = X[t];
    elsif PSTATE.EL == EL3 then
        PMCR_EL0 = X[t];

```

A.6.3 PMCEID0_ELO, Performance Monitors Common Event Identification register 0

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0000 to 0x001F and 0x4000 to 0x401F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. For more information about the common events and the use of the PMCEID<n>_ELO registers see 'The PMU event number space and common events'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-86: AArch64_pmceid0_el0 bit assignments

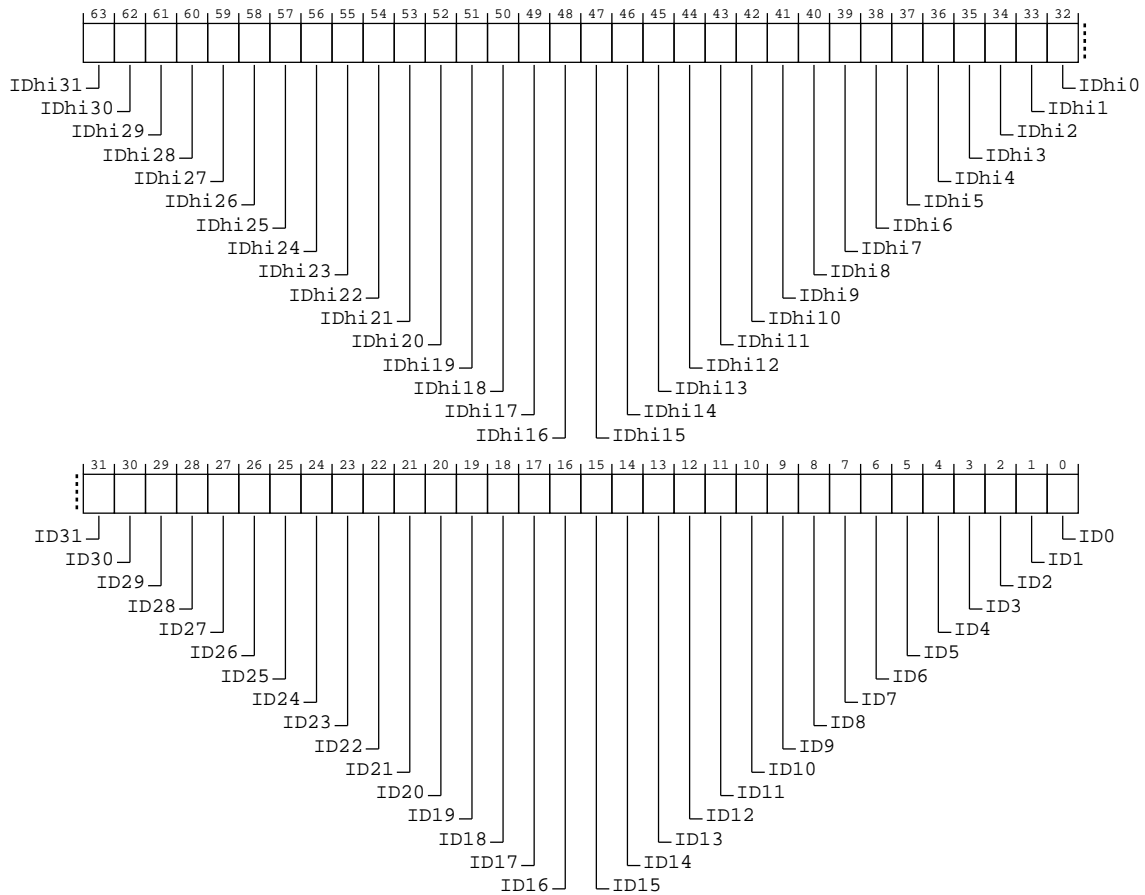


Table A-241: PMCEID0_ELO bit descriptions

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [63] | IDHi31 | IDHi31 corresponds to a Reserved Event event (0x401f) 0b0 The common event is not implemented, or not counted. | |
| [62] | IDHi30 | IDHi30 corresponds to a Reserved Event event (0x401e) 0b0 The common event is not implemented, or not counted. | |
| [61] | IDHi29 | IDHi29 corresponds to a Reserved Event event (0x401d) 0b0 The common event is not implemented, or not counted. | |
| [60] | IDHi28 | IDHi28 corresponds to a Reserved Event event (0x401c) 0b0 The common event is not implemented, or not counted. | |

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [59] | IDhi27 | IDhi27 corresponds to common event (0x401b) CTI_TRIGOUT7 0b1 The common event is implemented. | |
| [58] | IDhi26 | IDhi26 corresponds to common event (0x401a) CTI_TRIGOUT6 0b1 The common event is implemented. | |
| [57] | IDhi25 | IDhi25 corresponds to common event (0x4019) CTI_TRIGOUT5 0b1 The common event is implemented. | |
| [56] | IDhi24 | IDhi24 corresponds to common event (0x4018) CTI_TRIGOUT4 0b1 The common event is implemented. | |
| [55] | IDhi23 | IDhi23 corresponds to a Reserved Event event (0x4017) 0b0 The common event is not implemented, or not counted. | |
| [54] | IDhi22 | IDhi22 corresponds to a Reserved Event event (0x4016) 0b0 The common event is not implemented, or not counted. | |
| [53] | IDhi21 | IDhi21 corresponds to a Reserved Event event (0x4015) 0b0 The common event is not implemented, or not counted. | |
| [52] | IDhi20 | IDhi20 corresponds to a Reserved Event event (0x4014) 0b0 The common event is not implemented, or not counted. | |
| [51] | IDhi19 | IDhi19 corresponds to common event (0x4013) TRCEXTOUT3 0b1 The common event is implemented. | |
| [50] | IDhi18 | IDhi18 corresponds to common event (0x4012) TRCEXTOUT2 0b1 The common event is implemented. | |
| [49] | IDhi17 | IDhi17 corresponds to common event (0x4011) TRCEXTOUT1 0b1 The common event is implemented. | |
| [48] | IDhi16 | IDhi16 corresponds to common event (0x4010) TRCEXTOUT0 0b1 The common event is implemented. | |
| [47] | IDhi15 | IDhi15 corresponds to common event (0x400f) Reserved 0b0 The common event is not implemented, or not counted. | |
| [46] | IDhi14 | IDhi14 corresponds to common event (0x400e) TRB_TRIG 0b0 The common event is not implemented, or not counted. | |

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [45] | IDHi13 | IDHi13 corresponds to common event (0x400d) PMU_OVFS 0b0 The common event is not implemented, or not counted. | |
| [44] | IDHi12 | IDHi12 corresponds to common event (0x400c) TRB_WRAP 0b1 The common event is implemented. | |
| [43] | IDHi11 | IDHi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD 0b1 The common event is implemented. | |
| [42] | IDHi10 | IDHi10 corresponds to common event (0x400a) L2I_CACHE_LMISS 0b0 The common event is not implemented, or not counted. | |
| [41] | IDHi9 | IDHi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD 0b1 The common event is implemented. | |
| [40] | IDHi8 | IDHi8 corresponds to common event (0x4008) Reserved 0b0 The common event is not implemented, or not counted. | |
| [39] | IDHi7 | IDHi7 corresponds to common event (0x4007) Reserved 0b0 The common event is not implemented, or not counted. | |
| [38] | IDHi6 | IDHi6 corresponds to common event (0x4006) L1I_CACHE_LMISS 0b1 The common event is implemented. | |
| [37] | IDHi5 | IDHi5 corresponds to common event (0x4005) STALL_BACKEND_MEM 0b1 The common event is implemented. | |
| [36] | IDHi4 | IDHi4 corresponds to common event (0x4004) CNT_CYCLES 0b1 The common event is implemented. | |
| [35] | IDHi3 | IDHi3 corresponds to common event (0x4003) SAMPLE_COLLISION 0b1 The common event is implemented. | |
| [34] | IDHi2 | IDHi2 corresponds to common event (0x4002) SAMPLE_FILTRATE 0b1 The common event is implemented. | |
| [33] | IDHi1 | IDHi1 corresponds to common event (0x4001) SAMPLE_FEED 0b1 The common event is implemented. | |
| [32] | IDHi0 | IDHi0 corresponds to common event (0x4000) SAMPLE_POP 0b1 The common event is implemented. | |

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [31] | ID31 | ID31 corresponds to common event (0x1f) L1D_CACHE_ALLOCATE 0b0 The common event is not implemented, or not counted. | |
| [30] | ID30 | ID30 corresponds to common event (0x1e) CHAIN 0b1 The common event is implemented. | |
| [29] | ID29 | ID29 corresponds to common event (0x1d) BUS_CYCLES 0b1 The common event is implemented. | |
| [28] | ID28 | ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED 0b1 The common event is implemented. | |
| [27] | ID27 | ID27 corresponds to common event (0x1b) INST_SPEC 0b1 The common event is implemented. | |
| [26] | ID26 | ID26 corresponds to common event (0x1a) MEMORY_ERROR 0b1 The common event is implemented. | |
| [25] | ID25 | ID25 corresponds to common event (0x19) BUS_ACCESS 0b1 The common event is implemented. | |
| [24] | ID24 | ID24 corresponds to common event (0x18) L2D_CACHE_WB 0b1 The common event is implemented. | |
| [23] | ID23 | ID23 corresponds to common event (0x17) L2D_CACHE_REFILL 0b1 The common event is implemented. | |
| [22] | ID22 | ID22 corresponds to common event (0x16) L2D_CACHE 0b1 The common event is implemented. | |
| [21] | ID21 | ID21 corresponds to common event (0x15) L1D_CACHE_WB 0b1 The common event is implemented. | |
| [20] | ID20 | ID20 corresponds to common event (0x14) L1I_CACHE 0b1 The common event is implemented. | |
| [19] | ID19 | ID19 corresponds to common event (0x13) MEM_ACCESS 0b1 The common event is implemented. | |
| [18] | ID18 | ID18 corresponds to common event (0x12) BR_PRED 0b1 The common event is implemented. | |

| Bits | Name | Description | Reset |
|------|------|---|-------|
| [17] | ID17 | ID17 corresponds to common event (0x11) CPU_CYCLES 0b1 The common event is implemented. | |
| [16] | ID16 | ID16 corresponds to common event (0x10) BR_MIS_PRED 0b1 The common event is implemented. | |
| [15] | ID15 | ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [14] | ID14 | ID14 corresponds to common event (0xe) BR_RETURN_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [13] | ID13 | ID13 corresponds to common event (0xd) BR_IMMED_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [12] | ID12 | ID12 corresponds to common event (0xc) PC_WRITE_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [11] | ID11 | ID11 corresponds to common event (0xb) CID_WRITE_RETIRED 0b1 The common event is implemented. | |
| [10] | ID10 | ID10 corresponds to common event (0xa) EXC_RETURN 0b1 The common event is implemented. | |
| [9] | ID9 | ID9 corresponds to common event (0x9) EXC_TAKEN 0b1 The common event is implemented. | |
| [8] | ID8 | ID8 corresponds to common event (0x8) INST_RETIRED 0b1 The common event is implemented. | |
| [7] | ID7 | ID7 corresponds to common event (0x7) ST_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [6] | ID6 | ID6 corresponds to common event (0x6) LD_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [5] | ID5 | ID5 corresponds to common event (0x5) L1D_TLB_REFILL 0b1 The common event is implemented. | |
| [4] | ID4 | ID4 corresponds to common event (0x4) L1D_CACHE 0b1 The common event is implemented. | |

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [3] | ID3 | ID3 corresponds to common event (0x3) L1D_CACHE_REFILL 0b1 The common event is implemented. | |
| [2] | ID2 | ID2 corresponds to common event (0x2) L1I_TLB_REFILL 0b1 The common event is implemented. | |
| [1] | ID1 | ID1 corresponds to common event (0x1) L1I_CACHE_REFILL 0b1 The common event is implemented. | |
| [0] | ID0 | ID0 corresponds to common event (0x0) SW_INCR 0b1 The common event is implemented. | |

Access

MRS <Xt>, PMCEID0_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| PMCEID0_ELO | 0b11 | 0b011 | 0b1001 | 0b1100 | 0b110 |

Accessibility

MRS <Xt>, PMCEID0_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID0_ELO;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID0_ELO;
    elsif PSTATE.EL == EL2 then
        if MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID0_ELO;
    elsif PSTATE.EL == EL3 then
        return PMCEID0_ELO;

```

A.6.4 PMCEID1_ELO, Performance Monitors Common Event Identification register 1

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0020 to 0x003F and 0x4020 to 0x403F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. For more information about the common events and the use of the PMCEID<n>_ELO registers see 'The PMU event number space and common events'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-87: AArch64_pmceid1_el0 bit assignments

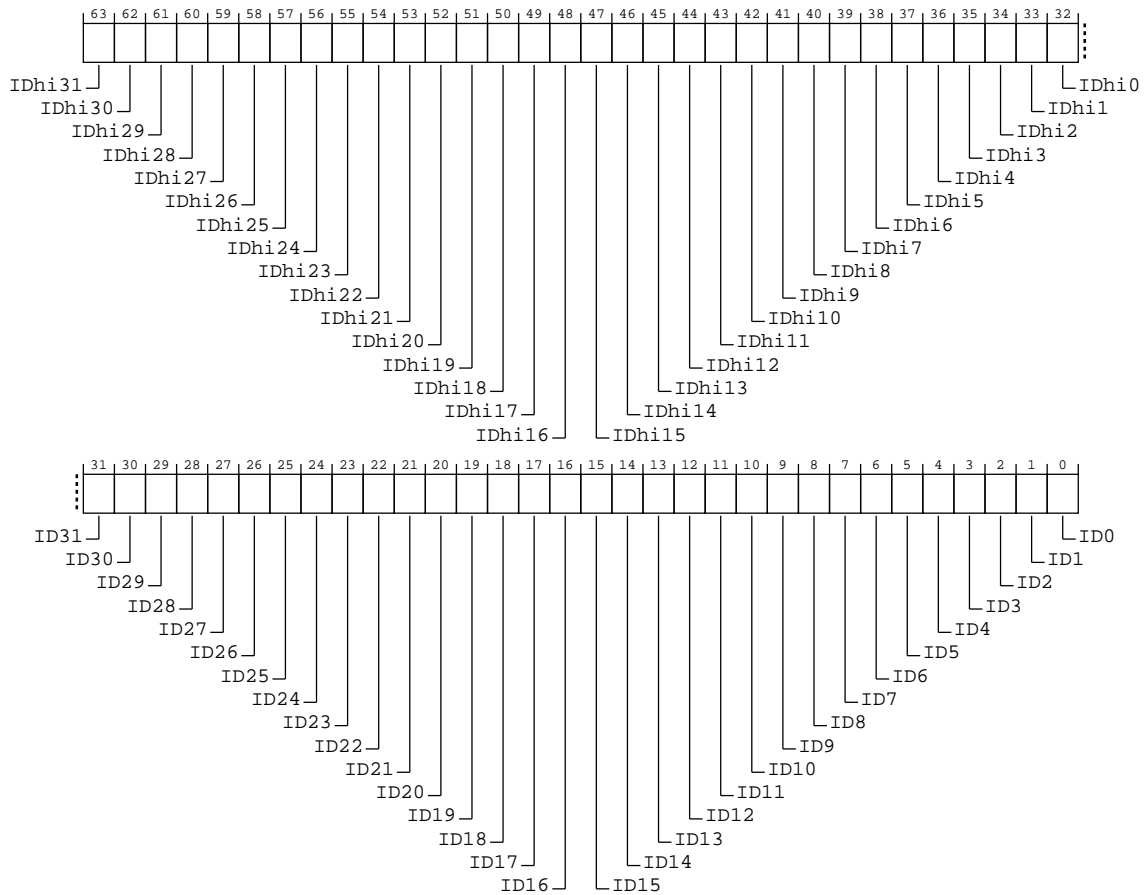


Table A-243: PMCEID1_ELO bit descriptions

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [63] | IDHi31 | IDHi31 corresponds to a Reserved Event event (0x403f) 0b0 The common event is not implemented, or not counted. | |
| [62] | IDHi30 | IDHi30 corresponds to a Reserved Event event (0x403e) 0b0 The common event is not implemented, or not counted. | |
| [61] | IDHi29 | IDHi29 corresponds to a Reserved Event event (0x403d) 0b0 The common event is not implemented, or not counted. | |
| [60] | IDHi28 | IDHi28 corresponds to a Reserved Event event (0x403c) 0b0 The common event is not implemented, or not counted. | |

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [59] | IDHi27 | IDHi27 corresponds to a Reserved Event event (0x403b) 0b0 The common event is not implemented, or not counted. | |
| [58] | IDHi26 | IDHi26 corresponds to a Reserved Event event (0x403a) 0b0 The common event is not implemented, or not counted. | |
| [57] | IDHi25 | IDHi25 corresponds to a Reserved Event event (0x4039) 0b0 The common event is not implemented, or not counted. | |
| [56] | IDHi24 | IDHi24 corresponds to a Reserved Event event (0x4038) 0b0 The common event is not implemented, or not counted. | |
| [55] | IDHi23 | IDHi23 corresponds to a Reserved Event event (0x4037) 0b0 The common event is not implemented, or not counted. | |
| [54] | IDHi22 | IDHi22 corresponds to a Reserved Event event (0x4036) 0b0 The common event is not implemented, or not counted. | |
| [53] | IDHi21 | IDHi21 corresponds to a Reserved Event event (0x4035) 0b0 The common event is not implemented, or not counted. | |
| [52] | IDHi20 | IDHi20 corresponds to a Reserved Event event (0x4034) 0b0 The common event is not implemented, or not counted. | |
| [51] | IDHi19 | IDHi19 corresponds to a Reserved Event event (0x4033) 0b0 The common event is not implemented, or not counted. | |
| [50] | IDHi18 | IDHi18 corresponds to a Reserved Event event (0x4032) 0b0 The common event is not implemented, or not counted. | |
| [49] | IDHi17 | IDHi17 corresponds to a Reserved Event event (0x4031) 0b0 The common event is not implemented, or not counted. | |
| [48] | IDHi16 | IDHi16 corresponds to a Reserved Event event (0x4030) 0b0 The common event is not implemented, or not counted. | |
| [47] | IDHi15 | IDHi15 corresponds to a Reserved Event event (0x402f) 0b0 The common event is not implemented, or not counted. | |
| [46] | IDHi14 | IDHi14 corresponds to a Reserved Event event (0x402e) 0b0 The common event is not implemented, or not counted. | |

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [45] | IDhi13 | IDhi13 corresponds to a Reserved Event event (0x402d) 0b0 The common event is not implemented, or not counted. | |
| [44] | IDhi12 | IDhi12 corresponds to a Reserved Event event (0x402c) 0b0 The common event is not implemented, or not counted. | |
| [43] | IDhi11 | IDhi11 corresponds to a Reserved Event event (0x402b) 0b0 The common event is not implemented, or not counted. | |
| [42] | IDhi10 | IDhi10 corresponds to a Reserved Event event (0x402a) 0b0 The common event is not implemented, or not counted. | |
| [41] | IDhi9 | IDhi9 corresponds to a Reserved Event event (0x4029) 0b0 The common event is not implemented, or not counted. | |
| [40] | IDhi8 | IDhi8 corresponds to a Reserved Event event (0x4028) 0b0 The common event is not implemented, or not counted. | |
| [39] | IDhi7 | IDhi7 corresponds to a Reserved Event event (0x4027) 0b0 The common event is not implemented, or not counted. | |
| [38] | IDhi6 | IDhi6 corresponds to common event (0x4026) MEM_ACCESS_CHECKED_WR 0b1 The common event is implemented. | |
| [37] | IDhi5 | IDhi5 corresponds to common event (0x4025) MEM_ACCESS_CHECKED_RD 0b1 The common event is implemented. | |
| [36] | IDhi4 | IDhi4 corresponds to common event (0x4024) MEM_ACCESS_CHECKED 0b1 The common event is implemented. | |
| [35] | IDhi3 | IDhi3 corresponds to common event (0x4023) Reserved 0b0 The common event is not implemented, or not counted. | |
| [34] | IDhi2 | IDhi2 corresponds to common event (0x4022) ST_ALIGN_LAT 0b1 The common event is implemented. | |
| [33] | IDhi1 | IDhi1 corresponds to common event (0x4021) LD_ALIGN_LAT 0b1 The common event is implemented. | |
| [32] | IDhi0 | IDhi0 corresponds to common event (0x4020) LDST_ALIGN_LAT 0b1 The common event is implemented. | |

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [31] | ID31 | ID31 corresponds to common event (0x3f) STALL_SLOT 0b1 The common event is implemented. | |
| [30] | ID30 | ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND 0b1 The common event is implemented. | |
| [29] | ID29 | ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND 0b1 The common event is implemented. | |
| [28] | ID28 | ID28 corresponds to common event (0x3c) STALL 0b1 The common event is implemented. | |
| [27] | ID27 | ID27 corresponds to common event (0x3b) OP_SPEC 0b1 The common event is implemented. | |
| [26] | ID26 | ID26 corresponds to common event (0x3a) OP_RETIRED 0b1 The common event is implemented. | |
| [25] | ID25 | ID25 corresponds to common event (0x39) L1D_CACHE_LMISS_RD 0b1 The common event is implemented. | |
| [24] | ID24 | ID24 corresponds to common event (0x38) REMOTE_ACCESS_RD 0b0 The common event is not implemented, or not counted. | |
| [23] | ID23 | ID23 corresponds to common event (0x37) LL_CACHE_MISS_RD 0b1 The common event is implemented. | |
| [22] | ID22 | ID22 corresponds to common event (0x36) LL_CACHE_RD 0b1 The common event is implemented. | |
| [21] | ID21 | ID21 corresponds to common event (0x35) ITLB_WALK 0b1 The common event is implemented. | |
| [20] | ID20 | ID20 corresponds to common event (0x34) DTLB_WALK 0b1 The common event is implemented. | |
| [19] | ID19 | ID19 corresponds to a Reserved Event event (0x33) 0b0 The common event is not implemented, or not counted. | |
| [18] | ID18 | ID18 corresponds to a Reserved Event event (0x32) 0b0 The common event is not implemented, or not counted. | |

| Bits | Name | Description | Reset |
|------|------|---|-------|
| [17] | ID17 | ID17 corresponds to common event (0x31) REMOTE_ACCESS 0b1 The common event is implemented. | |
| [16] | ID16 | ID16 corresponds to common event (0x30) L2I_TLB 0b0 The common event is not implemented, or not counted. | |
| [15] | ID15 | ID15 corresponds to common event (0x2f) L2D_TLB 0b1 The common event is implemented. | |
| [14] | ID14 | ID14 corresponds to common event (0x2e) L2I_TLB_REFILL 0b0 The common event is not implemented, or not counted. | |
| [13] | ID13 | ID13 corresponds to common event (0x2d) L2D_TLB_REFILL 0b1 The common event is implemented. | |
| [12] | ID12 | ID12 corresponds to common event (0x2c) Reserved 0b0 The common event is not implemented, or not counted. | |
| [11] | ID11 | ID11 corresponds to common event (0x2b) L3D_CACHE 0b1 The common event is implemented. | |
| [10] | ID10 | ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL 0b1 The common event is implemented. | |
| [9] | ID9 | ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE 0b1 The common event is implemented. | |
| [8] | ID8 | ID8 corresponds to common event (0x28) L2I_CACHE_REFILL 0b0 The common event is not implemented, or not counted. | |
| [7] | ID7 | ID7 corresponds to common event (0x27) L2I_CACHE 0b0 The common event is not implemented, or not counted. | |
| [6] | ID6 | ID6 corresponds to common event (0x26) L1I_TLB 0b1 The common event is implemented. | |
| [5] | ID5 | ID5 corresponds to common event (0x25) L1D_TLB 0b1 The common event is implemented. | |
| [4] | ID4 | ID4 corresponds to common event (0x24) STALL_BACKEND 0b1 The common event is implemented. | |

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [3] | ID3 | ID3 corresponds to common event (0x23) STALL_FRONTEND 0b1 The common event is implemented. | |
| [2] | ID2 | ID2 corresponds to common event (0x22) BR_MIS_PRED_RETIRED 0b1 The common event is implemented. | |
| [1] | ID1 | ID1 corresponds to common event (0x21) BR_RETIRED 0b1 The common event is implemented. | |
| [0] | ID0 | ID0 corresponds to common event (0x20) L2D_CACHE_ALLOCATE 0b1 The common event is implemented. | |

Access

MRS <Xt>, PMCEID1_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| PMCEID1_ELO | 0b11 | 0b011 | 0b1001 | 0b1100 | 0b111 |

Accessibility

MRS <Xt>, PMCEID1_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID1_ELO;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID1_ELO;
    elsif PSTATE.EL == EL2 then
        if MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID1_ELO;
    elsif PSTATE.EL == EL3 then
        return PMCEID1_ELO;

```

A.7 AArch64 GIC register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Generic Interrupt Controller (GIC) registers in the core. For more information about a register, you can click the register name in the table.

Table A-245: AArch64 GIC register summary

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

A.7.1 ICC_CTLR_EL1, Interrupt Controller Control Register (EL1)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC

Reset value

See individual bit resets

Bit descriptions

Figure A-88: AArch64_icc_ctlr_el1 bit assignments

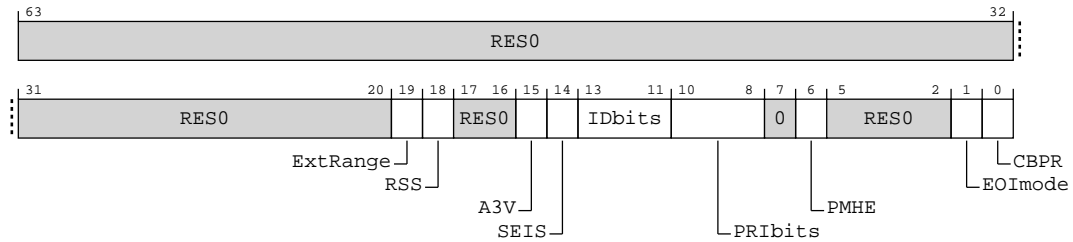


Table A-246: ICC_CTLR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [63:20] | RES0 | Reserved | 0x0 |
| [19] | ExtRange | Extended INTID range (read-only). 0b1 CPU interface supports INTIDs in the range 1024..8191 <ul style="list-style-type: none"> All INTIDs in the range 1024..8191 are treated as requiring deactivation. | |
| [18] | RSS | Range Selector Support. Possible values are: 0b0 Targeted SGIs with affinity level 0 values of 0-15 are supported. | |
| [17:16] | RES0 | Reserved | 0b00 |
| [15] | A3V | Affinity 3 Valid. Read-only and writes are ignored. Possible values are: 0b1 The CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers. | |
| [14] | SEIS | SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs: 0b0 The CPU interface logic does not support local generation of SEIs. | |
| [13:11] | IDbits | Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported: 0b000 16 bits. | |

| Bits | Name | Description | Reset |
|--------|---------|---|--------|
| [10:8] | PRIbits | <p>Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.</p> <p>An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).</p> <p>An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).</p> <p>Note: This field always returns the number of priority bits implemented, regardless of the Security state of the access or the value of ext-GICD_CTLR.DS. For physical accesses, this field determines the minimum value of AArch64-ICC_BPR0_EL1.</p> <p>If EL3 is implemented, physical accesses return the value from AArch64-ICC_CTLR_EL3.PRIbits.</p> <p>0b100 5 bits of priority are implemented</p> | |
| [7] | RES0 | Reserved | 0b0 |
| [6] | PMHE | <p>Priority Mask Hint Enable. Controls whether the priority mask register is used as a hint for interrupt distribution:</p> <p>0b0 Disables use of AArch64-ICC_PMR_EL1 as a hint for interrupt distribution.</p> <p>0b1 Enables use of AArch64-ICC_PMR_EL1 as a hint for interrupt distribution.</p> <p>If EL3 is implemented, this bit is an alias of AArch64-ICC_CTLR_EL3.PMHE. Whether this bit can be written as part of an access to this register depends on the value of ext-GICD_CTLR.DS:</p> <ul style="list-style-type: none"> If ext-GICD_CTLR.DS == 0, this bit is read-only. If ext-GICD_CTLR.DS == 1, this bit is read/write. | |
| [5:2] | RES0 | Reserved | 0b0000 |
| [1] | EOImode | <p>EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt:</p> <p>0b0 AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are UNPREDICTABLE.</p> <p>0b1 AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.</p> <p>The Secure AArch64-ICC_CTLR_EL1.EOImode is an alias of AArch64-ICC_CTLR_EL3.EOImode_EL1S.</p> <p>The Non-secure AArch64-ICC_CTLR_EL1.EOImode is an alias of AArch64-ICC_CTLR_EL3.EOImode_EL1NS</p> | |

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [0] | CBPR | <p>Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupts:</p> <p>0b0</p> <p>AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only.</p> <p>AArch64-ICC_BPR1_EL1 determines the preemption group for Group 1 interrupts.</p> <p>0b1</p> <p>AArch64-ICC_BPR0_EL1 determines the preemption group for both Group 0 and Group 1 interrupts.</p> <p>If EL3 is implemented:</p> <ul style="list-style-type: none"> This bit is an alias of AArch64-ICC_CTLR_EL3.CBPR_EL1{S,NS} where S or NS corresponds to the current Security state. If ext-GICD_CTLR.DS == 0, this bit is read-only. If ext-GICD_CTLR.DS == 1, this bit is read/write. | |

Access

MRS <Xt>, ICC_CTLR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ICC_CTLR_EL1 | 0b11 | 0b000 | 0b1100 | 0b1100 | 0b100 |

MSR ICC_CTLR_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ICC_CTLR_EL1 | 0b11 | 0b000 | 0b1100 | 0b1100 | 0b100 |

Accessibility

MRS <Xt>, ICC_CTLR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        return ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        return ICV_CTLR_EL1;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
    end if
elsif PSTATE.EL == EL2 then
    if SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
        end if
    end if
elsif PSTATE.EL == EL3 then

```

```

if SCR_EL3.NS == '0' then
    return ICC_CTLR_EL1_S;
else
    return ICC_CTLR_EL1_NS;

```

MSR ICC_CTLR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.<IRQ,FIQ> == '11' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t];
            else
                ICC_CTLR_EL1_NS = X[t];
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];

```

A.7.2 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register

Controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC

Reset value

See individual bit resets

Bit descriptions

Figure A-89: AArch64_icv_ctlr_el1 bit assignments

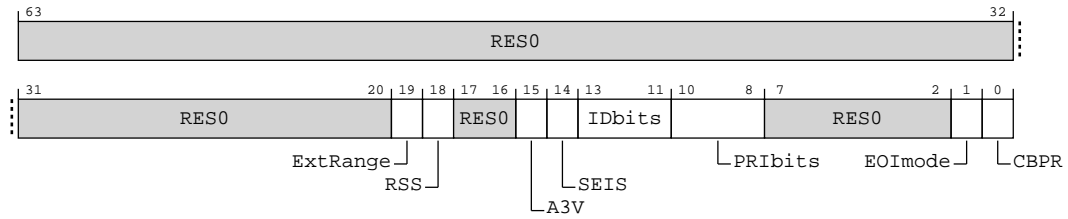


Table A-249: ICV_CTLR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|--|----------|
| [63:20] | RES0 | Reserved | 0x0 |
| [19] | ExtRange | Extended INTID range (read-only). 0b1 CPU interface supports INTIDs in the range 1024..8191 <ul style="list-style-type: none"> All INTIDs in the range 1024..8191 are treated as requiring deactivation. | |
| [18] | RSS | Range Selector Support. Possible values are: 0b0 Targeted SGIs with affinity level 0 values of 0-15 are supported. | |
| [17:16] | RES0 | Reserved | 0b00 |
| [15] | A3V | Affinity 3 Valid. Read-only and writes are ignored. Possible values are: 0b1 The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers. | |
| [14] | SEIS | SEI Support. Read-only and writes are ignored. Indicates whether the virtual CPU interface supports local generation of SEIs: 0b0 The virtual CPU interface logic does not support local generation of SEIs. | |
| [13:11] | IDbits | Identifier bits. Read-only and writes are ignored. The number of virtual interrupt identifier bits supported: 0b000 16 bits. | |
| [10:8] | PRIbits | Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation must implement at least 32 levels of physical priority (5 priority bits). Note: This field always returns the number of priority bits implemented. The division between group priority and subpriority is defined in the binary point registers AArch64-ICV_BPRO_EL1 and AArch64-ICV_BPR1_EL1. 0b100 5 bits of priority are implemented | |
| [7:2] | RES0 | Reserved | 0b000000 |

| Bits | Name | Description | Reset |
|------|---------|--|-------|
| [1] | EOImode | <p>Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt:</p> <p>0b0</p> <p>AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICV_DIR_EL1 are UNPREDICTABLE.</p> <p>0b1</p> <p>AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide priority drop functionality only. AArch64-ICV_DIR_EL1 provides interrupt deactivation functionality.</p> | |
| [0] | CBPR | <p>Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts:</p> <p>0b0</p> <p>AArch64-ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts.</p> <p>0b1</p> <p>Reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1 plus one, saturated to 0b111. Writes to AArch64-ICV_BPR1_EL1 are ignored.</p> | |

Access

MRS <Xt>, ICC_CTLR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ICC_CTLR_EL1 | 0b11 | 0b000 | 0b1100 | 0b1100 | 0b100 |

MSR ICC_CTLR_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ICC_CTLR_EL1 | 0b11 | 0b000 | 0b1100 | 0b1100 | 0b100 |

Accessibility

MRS <Xt>, ICC_CTLR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.FMO == '1' then
        return ICV_CTLR_EL1;
    elseif EL2Enabled() && HCR_EL2.IMO == '1' then
        return ICV_CTLR_EL1;
    elseif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
    elseif PSTATE.EL == EL2 then
        if SCR_EL3.<IRQ,FIQ> == '11' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            if SCR_EL3.NS == '0' then
                return ICC_CTLR_EL1_S;
            else

```

```

        return ICC_CTLR_EL1_NS;
elseif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        return ICC_CTLR_EL1_S;
    else
        return ICC_CTLR_EL1_NS;

```

MSR ICC_CTLR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elseif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elseif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
    elseif PSTATE.EL == EL2 then
        if SCR_EL3.<IRQ,FIQ> == '11' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t];
            else
                ICC_CTLR_EL1_NS = X[t];
    elseif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];

```

A.7.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Registers

Provides information about Group 0 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC

Reset value

See individual bit resets

Bit descriptions

Figure A-90: AArch64_icc_ap0r0_el1 bit assignments

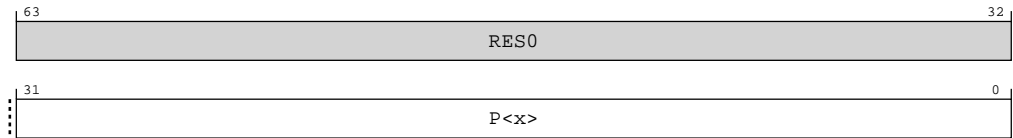


Table A-252: ICC_AP0R0_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|--|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:0] | P<x> | Provides the access to the active priorities for Group 0 interrupts. Possible values of each bit are: 0b0 There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. 0b1 There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3]. | |

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

MRS <Xt>, ICC_AP0R0_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ICC_AP0R0_EL1 | 0b11 | 0b000 | 0b1100 | 0b1000 | 0b100 |

MSR ICC_AP0R0_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ICC_AP0R0_EL1 | 0b11 | 0b000 | 0b1100 | 0b1000 | 0b100 |

A.7.4 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers

Provides information about virtual Group 0 active priorities.

Configurations

This register is available in all configurations.

Attributes**Width**

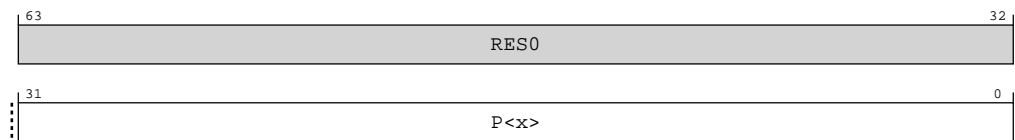
64

Functional group

GIC

Reset value

See individual bit resets

Bit descriptions**Figure A-91: AArch64_icv_ap0r0_el1 bit assignments****Table A-255: ICV_AP0R0_EL1 bit descriptions**

| Bits | Name | Description | Reset |
|---------|------|--|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:0] | P<x> | Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: 0b0 There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. 0b1 There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3]. | |

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

MRS <Xt>, ICC_AP0R0_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ICC_AP0R0_EL1 | 0b11 | 0b000 | 0b1100 | 0b1000 | 0b100 |

MSR ICC_AP0R0_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ICC_AP0R0_EL1 | 0b11 | 0b000 | 0b1100 | 0b1000 | 0b100 |

A.7.5 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Registers

Provides information about Group 1 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC

Reset value

See individual bit resets

Bit descriptions

Figure A-92: AArch64_icc_ap1r0_el1 bit assignments

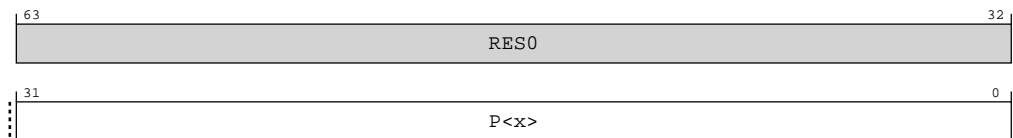


Table A-258: ICC_AP1R0_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|---|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:0] | P<x> | <p>Group 1 interrupt active priorities. When AArch64-SCR_EL3.NS == '1', accesses the priorities for Non-secure Group 1 interrupts, and when AArch64-SCR_EL3.NS == '0' accesses the priorities for Secure Group 1 interrupts. Possible values of each bit are:</p> <p>0b0</p> <p>There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop.</p> <p>0b1</p> <p>There is a Group 1 interrupt active with this priority level which has not undergone priority drop.</p> <p>There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].</p> <p>When accessed from non-secure EL2 or EL1, only the 16 lowest-priority interrupts are visible in bits [15:0] of this register.</p> | |

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

MRS <Xt>, ICC_AP1R0_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ICC_AP1R0_EL1 | 0b11 | 0b000 | 0b1100 | 0b1001 | 0b000 |

MSR ICC_AP1R0_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ICC_AP1R0_EL1 | 0b11 | 0b000 | 0b1100 | 0b1001 | 0b000 |

A.7.6 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers

Provides information about virtual Group 1 active priorities.

Configurations

This register is available in all configurations.

Attributes**Width**

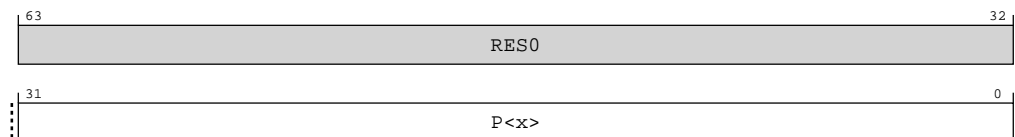
64

Functional group

GIC

Reset value

See individual bit resets

Bit descriptions**Figure A-93: AArch64_icv_ap1r0_el1 bit assignments****Table A-261: ICV_AP1R0_EL1 bit descriptions**

| Bits | Name | Description | Reset |
|---------|------|-------------|-------|
| [63:32] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|--------|------|---|-------|
| [31:0] | P<x> | <p>Group 1 interrupt active priorities. Possible values of each bit are:</p> <p>0b0</p> <p>There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop.</p> <p>0b1</p> <p>There is a Group 1 interrupt active with this priority level which has not undergone priority drop.</p> <p>There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].</p> | |

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

MRS <Xt>, ICC_AP1R0_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ICC_AP1R0_EL1 | 0b11 | 0b000 | 0b1100 | 0b1001 | 0b000 |

MSR ICC_AP1R0_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ICC_AP1R0_EL1 | 0b11 | 0b000 | 0b1100 | 0b1001 | 0b000 |

A.7.7 ICH_VTR_EL2, Interrupt Controller VGIC Type Register

Reports supported GIC virtualization features.

Configurations

If EL2 is not implemented, all bits in this register are RES0 from EL3, except for nV4, which is RES1 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

GIC

Reset value

See individual bit resets

Bit descriptions

Figure A-94: AArch64_ich_vtr_el2 bit assignments

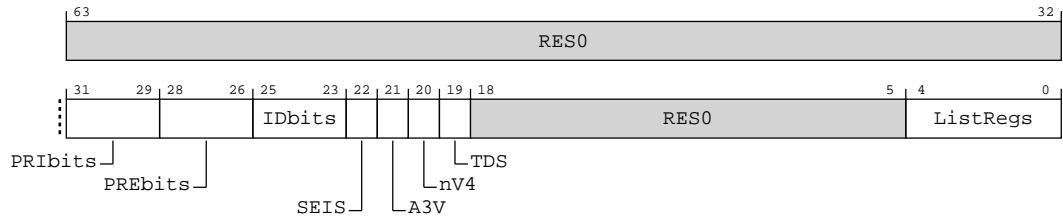


Table A-264: ICH_VTR_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|---------|---|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:29] | PRIbits | <p>Priority bits. The number of virtual priority bits implemented, minus one.</p> <p>An implementation must implement at least 32 levels of virtual priority (5 priority bits).</p> <p>This field is an alias of AArch64-ICV_CTLR_EL1.PRIbits.</p> <p>0b100</p> <p>5 virtual priority bits are implemented</p> | |
| [28:26] | PREbits | <p>The number of virtual preemption bits implemented, minus one.</p> <p>An implementation must implement at least 32 levels of virtual preemption priority (5 preemption bits).</p> <p>The value of this field must be less than or equal to the value of ICH_VTR_EL2.PRIbits.</p> <p>The maximum value of this field is 6, indicating 7 bits of preemption.</p> <p>This field determines the minimum value of AArch64-ICH_VMCR_EL2.VBPR0.</p> <p>0b100</p> <p>5 virtual preemption bits are implemented</p> | |
| [25:23] | IDbits | <p>The number of virtual interrupt identifier bits supported:</p> <p>0b000</p> <p>16 bits.</p> | |
| [22] | SEIS | <p>SEI Support. Indicates whether the virtual CPU interface supports generation of SEIs:</p> <p>0b0</p> <p>The virtual CPU interface logic does not support generation of SEIs.</p> | |
| [21] | A3V | <p>Affinity 3 Valid. Possible values are:</p> <p>0b1</p> <p>The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.</p> | |
| [20] | nV4 | <p>Direct injection of virtual interrupts not supported. Possible values are:</p> <p>0b0</p> <p>The CPU interface logic supports direct injection of virtual interrupts.</p> | |

| Bits | Name | Description | Reset |
|--------|----------|---|-------|
| [19] | TDS | Separate trapping of EL1 writes to AArch64-ICV_DIR_EL1 supported. 0b1 Implementation supports AArch64-ICH_HCR_EL2.TDIR. | |
| [18:5] | RES0 | Reserved | 0x0 |
| [4:0] | ListRegs | The number of implemented List registers, minus one. For example, a value of 0b01111 indicates that the maximum of 16 List registers are implemented. 0b00011 4 List registers | |

Access

MRS <Xt>, ICH_VTR_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ICH_VTR_EL2 | 0b11 | 0b100 | 0b1100 | 0b1011 | 0b001 |

Accessibility

MRS <Xt>, ICH_VTR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    return ICH_VTR_EL2;
elseif PSTATE.EL == EL3 then
    return ICH_VTR_EL2;

```

A.7.8 ICC_CTLR_EL3, Interrupt Controller Control Register (EL3)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC

Reset value

See individual bit resets

Bit descriptions

Figure A-95: AArch64_icc_ctlr_el3 bit assignments

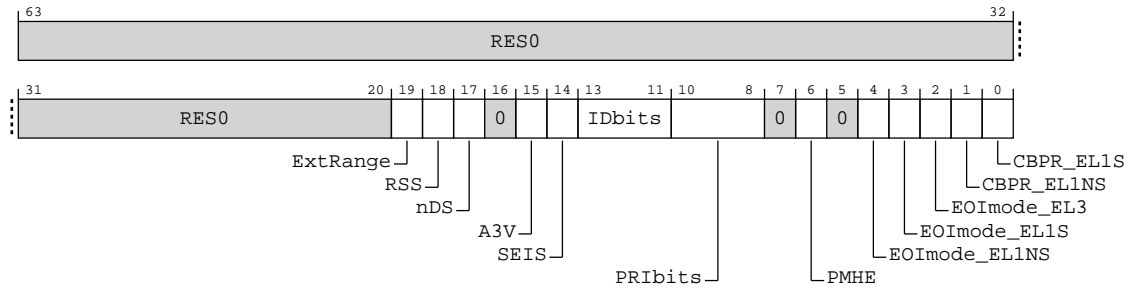


Table A-266: ICC_CTLR_EL3 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [63:20] | RES0 | Reserved | 0x0 |
| [19] | ExtRange | Extended INTID range (read-only). 0b1 CPU interface supports INTIDs in the range 1024..8191 <ul style="list-style-type: none"> All INTIDs in the range 1024..8191 are treated as requiring deactivation. | |
| [18] | RSS | Range Selector Support. 0b0 Targeted SGIs with affinity level 0 values of 0-15 are supported. | |
| [17] | nDS | Disable Security not supported. Read-only and writes are ignored. 0b1 The CPU interface logic does not support disabling of security, and requires that security is not disabled. | |
| [16] | RES0 | Reserved | 0x0 |
| [15] | A3V | Affinity 3 Valid. Read-only and writes are ignored. 0b1 The CPU interface logic supports non-zero values of the Aff3 field in SGI generation System registers. | |
| [14] | SEIS | SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports generation of SEIs: 0b0 The CPU interface logic does not support generation of SEIs. | |
| [13:11] | IDbits | Identifier bits. Read-only and writes are ignored. Indicates the number of physical interrupt identifier bits supported. 0b000 16 bits. | |

| Bits | Name | Description | Reset |
|--------|---------------|--|-------|
| [10:8] | PRIbits | <p>Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.</p> <p>An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).</p> <p>An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).</p> <p>Note: This field always returns the number of priority bits implemented, regardless of the value of SCR_EL3.NS or the value of ext-GICD_CTLR.DS. The division between group priority and subpriority is defined in the binary point registers AArch64-ICC_BPR0_EL1 and AArch64-ICC_BPR1_EL1.</p> <p>This field determines the minimum value of ICC_BPR0_EL1.</p> <p>0b100 5 bits of priority are implemented</p> | |
| [7] | RES0 | Reserved | 0b0 |
| [6] | PMHE | <p>Priority Mask Hint Enable.</p> <p>0b0 Disables use of the priority mask register as a hint for interrupt distribution.</p> <p>0b1 Enables use of the priority mask register as a hint for interrupt distribution.</p> <p>Software must write AArch64-ICC_PMR_EL1 to 0xFF before clearing this field to 0.</p> <ul style="list-style-type: none"> An implementation might choose to make this field RAO/WI if priority-based routing is always used An implementation might choose to make this field RAZ/WI if priority-based routing is never used <p>If EL3 is present, AArch64-ICC_CTLR_EL1.PMHE is an alias of ICC_CTLR_EL3.PMHE.</p> | 0b0 |
| [5] | RES0 | Reserved | 0b0 |
| [4] | EOImode_EL1NS | <p>EOI mode for interrupts handled at Non-secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt.</p> <p>0b0 AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are UNPREDICTABLE.</p> <p>0b1 AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.</p> <p>If EL3 is present, AArch64-ICC_CTLR_EL1(NS).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1NS.</p> | |

| Bits | Name | Description | Reset |
|------|--------------|---|-------|
| [3] | EOImode_EL1S | <p>EOI mode for interrupts handled at Secure EL1. Controls whether a write to an End of Interrupt register also deactivates the interrupt.</p> <p>0b0</p> <p>AArch64-ICC_EOIRO_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are UNPREDICTABLE.</p> <p>0b1</p> <p>AArch64-ICC_EOIRO_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.</p> <p>If EL3 is present, AArch64-ICC_CTLR_EL1(S).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1S.</p> | |
| [2] | EOImode_EL3 | <p>EOI mode for interrupts handled at EL3. Controls whether a write to an End of Interrupt register also deactivates the interrupt.</p> <p>0b0</p> <p>AArch64-ICC_EOIRO_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are UNPREDICTABLE.</p> <p>0b1</p> <p>AArch64-ICC_EOIRO_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.</p> | |
| [1] | CBPR_EL1NS | <p>Common Binary Point Register, EL1 Non-secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Non-secure interrupts at EL1 and EL2.</p> <p>0b0</p> <p>AArch64-ICC_BPRO_EL1 determines the preemption group for Group 0 interrupts only.</p> <p>AArch64-ICC_BPR1_EL1 determines the preemption group for Non-secure Group 1 interrupts.</p> <p>0b1</p> <p>AArch64-ICC_BPRO_EL1 determines the preemption group for Group 0 interrupts and Non-secure Group 1 interrupts. Non-secure accesses to ext-GICC_BPR and AArch64-ICC_BPR1_EL1 access the state of AArch64-ICC_BPRO_EL1.</p> <p>If EL3 is present, AArch64-ICC_CTLR_EL1(NS).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1NS.</p> | |
| [0] | CBPR_EL1S | <p>Common Binary Point Register, EL1 Secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Secure interrupts at EL1.</p> <p>0b0</p> <p>AArch64-ICC_BPRO_EL1 determines the preemption group for Group 0 interrupts only.</p> <p>AArch64-ICC_BPR1_EL1 determines the preemption group for Secure Group 1 interrupts.</p> <p>0b1</p> <p>AArch64-ICC_BPRO_EL1 determines the preemption group for Group 0 interrupts and Secure Group 1 interrupts. Secure EL1 accesses to AArch64-ICC_BPR1_EL1 access the state of AArch64-ICC_BPRO_EL1.</p> <p>If EL3 is present, AArch64-ICC_CTLR_EL1(S).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1S.</p> | |

Access

MRS <Xt>, ICC_CTLR_EL3

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ICC_CTLR_EL3 | 0b11 | 0b110 | 0b1100 | 0b1100 | 0b100 |

MSR ICC_CTLR_EL3, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ICC_CTLR_EL3 | 0b11 | 0b110 | 0b1100 | 0b1100 | 0b100 |

Accessibility

MRS <Xt>, ICC_CTLR_EL3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return ICC_CTLR_EL3;

```

MSR ICC_CTLR_EL3, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    ICC_CTLR_EL3 = X[t];

```

A.8 AArch64 Activity monitors register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** activity monitors registers in the core. For more information about a register, you can click the register name in the table.

Table A-269: AArch64 Activity monitors register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|---------------------------------|-----|-----|-----|-----|-----|---------------------------|--------|--|
| AMEVTYPER10_ELO | 3 | C13 | 3 | C14 | 0 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 1 |
| AMEVTYPER11_ELO | 3 | C13 | 3 | C14 | 1 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 1 |
| AMEVTYPER12_ELO | 3 | C13 | 3 | C14 | 2 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 1 |
| AMCFGR_ELO | 3 | C13 | 3 | C2 | 1 | See individual bit resets | 64-bit | Activity Monitors Configuration Register |
| AMCGCR_ELO | 3 | C13 | 3 | C2 | 2 | See individual bit resets | 64-bit | Activity Monitors Counter Group Configuration Register |

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-----------------|-----|-----|-----|-----|-----|---------------------------|--------|--|
| AMEVTYPER00_ELO | 3 | C13 | 3 | C6 | 0 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 0 |
| AMEVTYPER01_ELO | 3 | C13 | 3 | C6 | 1 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 0 |
| AMEVTYPER02_ELO | 3 | C13 | 3 | C6 | 2 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 0 |
| AMEVTYPER03_ELO | 3 | C13 | 3 | C6 | 3 | See individual bit resets | 64-bit | Activity Monitors Event Type Registers 0 |

A.8.1 AMEVTYPER10_ELO, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR10_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-96: AArch64_amevtyper10_el0 bit assignments

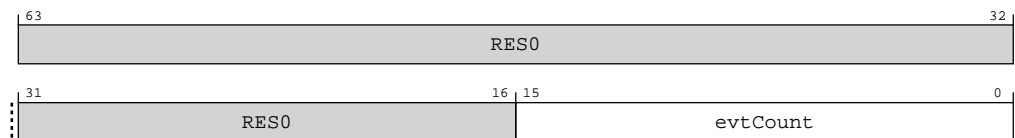


Table A-270: AMEVTYPER10_ELO bit descriptions

| Bits | Name | Description | Reset |
|---------|------|-------------|-------|
| [63:16] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|--------|----------|---|-------|
| [15:0] | evtCount | <p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AArch64-AMEVCNTR1<n>_ELO.</p> <p>It is IMPLEMENTATION DEFINED what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter AArch64-AMEVCNTR1<n>_ELO, then:</p> <ul style="list-style-type: none"> It is UNPREDICTABLE which event will be counted. The value read back is UNKNOWN. <p>The event counted by AArch64-AMEVCNTR1<n>_ELO might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED.</p> <p>If the corresponding counter AArch64-AMEVCNTR1<n>_ELO is enabled, writes to this register have UNPREDICTABLE results.</p> | |

Access

MRS <Xt>, AMEVTYPER10_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| AMEVTYPER10_ELO | 0b11 | 0b011 | 0b1101 | 0b1110 | 0b000 |

MSR AMEVTYPER10_ELO, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| AMEVTYPER10_ELO | 0b11 | 0b011 | 0b1101 | 0b1110 | 0b000 |

A.8.2 AMEVTYPER11_ELO, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR11_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-97: AArch64_amevtyper11_el0 bit assignments

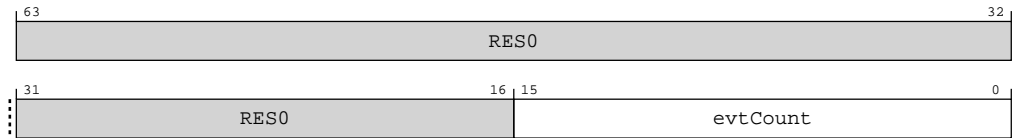


Table A-273: AMEVTYPER11_ELO bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [63:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | <p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AArch64-AMEVCNTR1<n>_ELO.</p> <p>It is IMPLEMENTATION DEFINED what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter AArch64-AMEVCNTR1<n>_ELO, then:</p> <ul style="list-style-type: none"> It is UNPREDICTABLE which event will be counted. The value read back is UNKNOWN. <p>The event counted by AArch64-AMEVCNTR1<n>_ELO might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED.</p> <p>If the corresponding counter AArch64-AMEVCNTR1<n>_ELO is enabled, writes to this register have UNPREDICTABLE results.</p> | |

Access

MRS <Xt>, AMEVTYPER11_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| AMEVTYPER11_ELO | 0b11 | 0b011 | 0b1101 | 0b1110 | 0b001 |

MSR AMEVTYPER11_ELO, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| AMEVTYPER11_ELO | 0b11 | 0b011 | 0b1101 | 0b1110 | 0b001 |

A.8.3 AMEVTYPER12_ELO, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR12_ELO counts.

Configurations

This register is available in all configurations.

Attributes**Width**

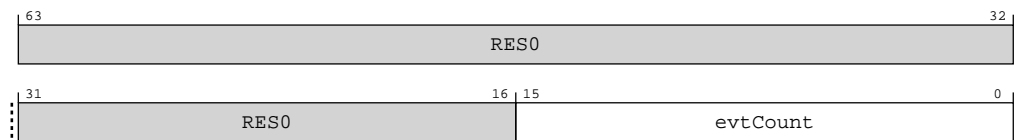
64

Functional group

Activity monitors

Reset value

See individual bit resets

Bit descriptions**Figure A-98: AArch64_amevtyper12_el0 bit assignments****Table A-276: AMEVTYPER12_ELO bit descriptions**

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [63:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | <p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AArch64-AMEVCNTR1<n>_ELO.</p> <p>It is IMPLEMENTATION DEFINED what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter AArch64-AMEVCNTR1<n>_ELO, then:</p> <ul style="list-style-type: none"> It is UNPREDICTABLE which event will be counted. The value read back is UNKNOWN. <p>The event counted by AArch64-AMEVCNTR1<n>_ELO might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED.</p> <p>If the corresponding counter AArch64-AMEVCNTR1<n>_ELO is enabled, writes to this register have UNPREDICTABLE results.</p> | |

Access

MRS <Xt>, AMEVTYPER12_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| AMEVTYPER12_ELO | 0b11 | 0b011 | 0b1101 | 0b1110 | 0b010 |

MSR AMEVTYPER12_ELO, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| AMEVTYPER12_ELO | 0b11 | 0b011 | 0b1101 | 0b1110 | 0b010 |

A.8.4 AMCFGR_ELO, Activity Monitors Configuration Register

Global configuration register for the activity monitors.

Provides information on supported features, the number of counter groups implemented, the total number of activity monitor event counters implemented, and the size of the counters. AMCFGR_ELO is applicable to both the architected and the auxiliary counter groups.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-99: AArch64_amcfgr_el0 bit assignments

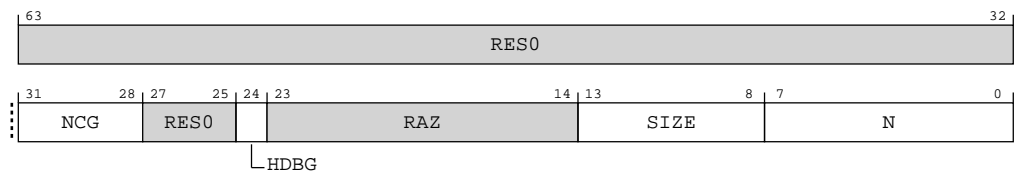


Table A-279: AMCFGR_ELO bit descriptions

| Bits | Name | Description | Reset |
|---------|------|---|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:28] | NCG | Defines the number of counter groups. The following value is specified for this product. 0b0001 Two counter groups are implemented | |
| [27:25] | RES0 | Reserved | 0b000 |
| [24] | HDBG | Halt-on-debug supported. From Armv8, this feature must be supported, and so this bit is 0b1. 0b1 AArch64-AMCR_ELO.HDBG is read/write. | |
| [23:14] | RAZ | Reserved | |

| Bits | Name | Description | Reset |
|--------|------|---|-------|
| [13:8] | SIZE | <p>Defines the size of activity monitor event counters.</p> <p>The size of the activity monitor event counters implemented by the activity monitors Extension is defined as [AMCFGR_ELO.SIZE + 1].</p> <p>From Armv8, the counters are 64-bit, and so this field is 0b111111.</p> <p>Note: Software also uses this field to determine the spacing of counters in the memory-map. From Armv8, the counters are at doubleword-aligned addresses.</p> <p>0b111111 64 bits.</p> | |
| [7:0] | N | <p>Defines the number of activity monitor event counters.</p> <p>The total number of counters implemented in all groups by the Activity Monitors Extension is defined as [AMCFGR_ELO.N + 1].</p> <p>0b00000110 Seven activity monitor event counters</p> | |

Access

MRS <Xt>, AMCFGR_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AMCFGR_ELO | 0b11 | 0b011 | 0b1101 | 0b0010 | 0b001 |

Accessibility

MRS <Xt>, AMCFGR_ELO

```

if PSTATE.EL == EL0 then
    if AMUSERENR_ELO.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCFGR_ELO;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCFGR_ELO;
    elsif PSTATE.EL == EL2 then
        if CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCFGR_ELO;
    elsif PSTATE.EL == EL3 then
        return AMCFGR_ELO;

```

A.8.5 AMCGCR_ELO, Activity Monitors Counter Group Configuration Register

Provides information on the number of activity monitor event counters implemented within each counter group.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-100: AArch64_amcgr_el0 bit assignments

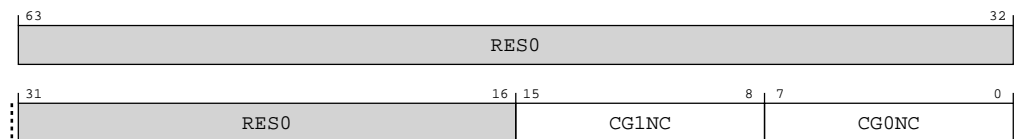


Table A-281: AMCGCR_ELO bit descriptions

| Bits | Name | Description | Reset |
|---------|-------|---|-------|
| [63:16] | RES0 | Reserved | 0x0 |
| [15:8] | CG1NC | Counter Group 1 Number of Counters. The number of counters in the auxiliary counter group. In AMUv1, the permitted range of values is 0x0 to 0x10. 0b00000011 Three counters in the auxiliary counter group | |
| [7:0] | CG0NC | Counter Group 0 Number of Counters. The number of counters in the architected counter group. In AMUv1, the value of this field is 0x4. 0b00000100 Four Counters in the architected counter group | |

Access

MRS <Xt>, AMCGCR_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| AMCGCR_ELO | 0b11 | 0b011 | 0b1101 | 0b0010 | 0b010 |

Accessibility

MRS <Xt>, AMCGCR_ELO

```

if PSTATE.EL == EL0 then
    if AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCGCR_EL0;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCGCR_EL0;
    elsif PSTATE.EL == EL2 then
        if CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCGCR_EL0;
    elsif PSTATE.EL == EL3 then
        return AMCGCR_EL0;

```

A.8.6 AMEVTYPER00_ELO, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR00_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-101: AArch64_amevtyper00_el0 bit assignments

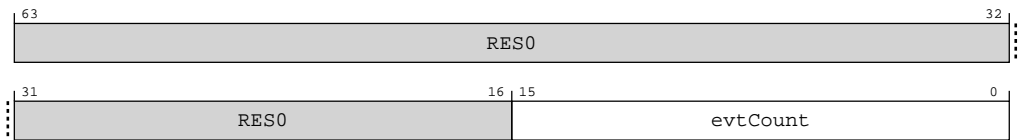


Table A-283: AMEVTYPER00_ELO bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [63:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR00_ELO. The value of this field is architecturally required for each architected counter. | |

Access

MRS <Xt>, AMEVTYPER00_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| AMEVTYPER00_ELO | 0b11 | 0b011 | 0b1101 | 0b0110 | 0b000 |

A.8.7 AMEVTYPER01_ELO, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR01_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-102: AArch64_amevtyper01_el0 bit assignments

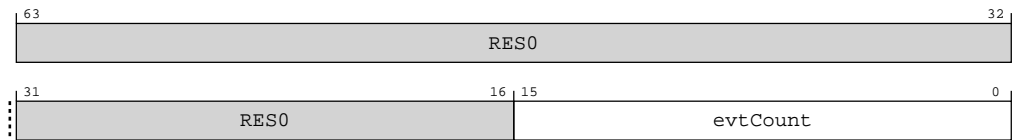


Table A-285: AMEVTYPER01_ELO bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [63:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR01_ELO. The value of this field is architecturally required for each architected counter. | |

Access

MRS <Xt>, AMEVTYPER01_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| AMEVTYPER01_ELO | 0b11 | 0b011 | 0b1101 | 0b0110 | 0b001 |

A.8.8 AMEVTYPER02_ELO, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR02_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-103: AArch64_amevtyper02_el0 bit assignments

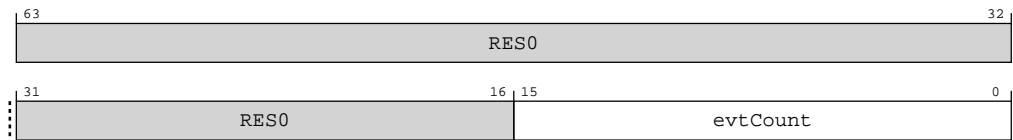


Table A-287: AMEVTYPER02_ELO bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [63:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR02_ELO. The value of this field is architecturally required for each architected counter. | |

Access

MRS <Xt>, AMEVTYPER02_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| AMEVTYPER02_ELO | 0b11 | 0b011 | 0b1101 | 0b0110 | 0b010 |

A.8.9 AMEVTYPER03_ELO, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR03_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity monitors

Reset value

See individual bit resets

Bit descriptions

Figure A-104: AArch64_amevtyper03_el0 bit assignments

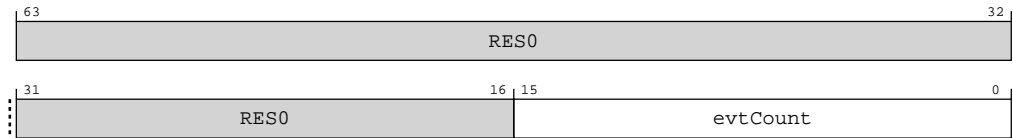


Table A-289: AMEVTYPER03_ELO bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [63:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR03_ELO. The value of this field is architecturally required for each architected counter. | |

Access

MRS <Xt>, AMEVTYPER03_ELO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-----------------|------|-------|--------|--------|-------|
| AMEVTYPER03_ELO | 0b11 | 0b011 | 0b1101 | 0b0110 | 0b011 |

A.9 AArch64 Trace register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** trace registers in the core. For more information about a register, you can click the register name in the table.

Table A-291: AArch64 Trace register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|----------------------------|-----|-----|-----|-----|-----|---------------------------|--------|---|
| TRCIDR8 | 2 | C0 | 1 | C0 | 6 | See individual bit resets | 64-bit | ID Register 8 |
| TRCIMSPECO | 2 | C0 | 1 | C0 | 7 | See individual bit resets | 64-bit | IMP DEF Register 0 |
| TRCIDR2 | 2 | C0 | 1 | C10 | 7 | See individual bit resets | 64-bit | ID Register 2 |
| TRCIDR3 | 2 | C0 | 1 | C11 | 7 | See individual bit resets | 64-bit | ID Register 3 |
| TRCIDR4 | 2 | C0 | 1 | C12 | 7 | See individual bit resets | 64-bit | ID Register 4 |
| TRCIDR5 | 2 | C0 | 1 | C13 | 7 | See individual bit resets | 64-bit | ID Register 5 |
| TRCIDR10 | 2 | C0 | 1 | C2 | 6 | 0x0 | 64-bit | ID Register 10 |
| TRCIDR11 | 2 | C0 | 1 | C3 | 6 | 0x0 | 64-bit | ID Register 11 |
| TRCIDR12 | 2 | C0 | 1 | C4 | 6 | 0x0 | 64-bit | ID Register 12 |
| TRCIDR13 | 2 | C0 | 1 | C5 | 6 | 0x0 | 64-bit | ID Register 13 |
| TRCIDR0 | 2 | C0 | 1 | C8 | 7 | See individual bit resets | 64-bit | ID Register 0 |
| TRCIDR1 | 2 | C0 | 1 | C9 | 7 | See individual bit resets | 64-bit | ID Register 1 |
| TRCCIDCVRO | 2 | C3 | 1 | C0 | 0 | See individual bit resets | 64-bit | Context Identifier Comparator Value Registers <n> |

A.9.1 TRCIDR8, ID Register 8

Returns the maximum speculation depth of the instruction trace element stream.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

See individual bit resets

Bit descriptions

Figure A-105: AArch64_trcidr8 bit assignments

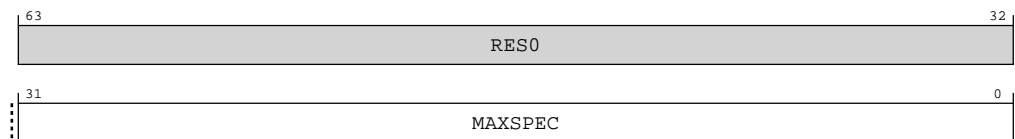


Table A-292: TRCIDR8 bit descriptions

| Bits | Name | Description | Reset |
|---------|---------|---|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:0] | MAXSPEC | Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of PO elements in the trace element stream that can be speculative at any time. 0x0 No speculation in the trace element stream | |

Access

MRS <Xt>, TRCIDR8

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIDR8 | 0b10 | 0b001 | 0b0000 | 0b0000 | 0b110 |

Accessibility

MRS <Xt>, TRCIDR8

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```

```

if CPACR_EL1.TTA == '1' then
    AArch64.SystemAccessTrap(EL1, 0x18);
elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elsif CPTR_EL3.TTA == '1' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    return TRCIDR8;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR8;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR8;

```

A.9.2 TRCIMSPECO, IMP DEF Register 0

TRCIMSPECO shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

See individual bit resets

Bit descriptions

Figure A-106: AArch64_trcimspec0 bit assignments

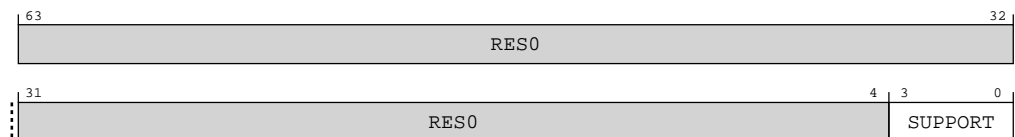


Table A-294: TRCIMSPECO bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|--|-------|
| [63:4] | RES0 | Reserved | 0x0 |
| [3:0] | SUPPORT | Indicates whether the implementation supports IMPLEMENTATION DEFINED features. 0b0000 No IMPLEMENTATION DEFINED features are supported. | |

Access

MRS <Xt>, TRCIMSPECO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIMSPECO | 0b10 | 0b001 | 0b0000 | 0b0000 | 0b111 |

MSR TRCIMSPECO, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIMSPECO | 0b10 | 0b001 | 0b0000 | 0b0000 | 0b111 |

Accessibility

MRS <Xt>, TRCIMSPECO

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCIMSPECN == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPECO;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPECO;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPECO;

```

MSR TRCIMSPECO, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then

```

```

    AArch64.SystemAccessTrap(EL2, 0x18);
elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDBGWTR_EL2.TRCIMSPECn == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elseif CPTR_EL3.TTA == '1' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    TRCIMSPECO = X[t];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPECO = X[t];
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPECO = X[t];

```

A.9.3 TRCIDR2, ID Register 2

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

See individual bit resets

Bit descriptions

Figure A-107: AArch64_trcidr2 bit assignments

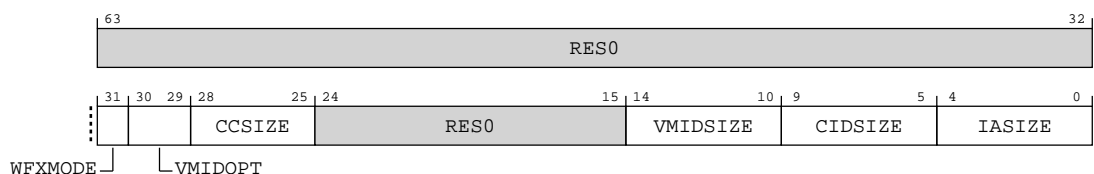


Table A-297: TRCIDR2 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|-------------|-------|
| [63:32] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [31] | WFXMODE | Indicates whether WFI and WFE instructions are classified as P0 instructions: 0b1 WFI and WFE instructions are classified as P0 instructions. | |
| [30:29] | VMIDOPT | Indicates the options for Virtual context identifier selection. 0b10 Virtual context identifier selection not supported. AArch64-TRCCONFIGR.VMIDOPT is RES1. | |
| [28:25] | CCSIZE | Indicates the size of the cycle counter. 0b0000 The cycle counter is 12 bits in length. | |
| [24:15] | RES0 | Reserved | 0x0 |
| [14:10] | VMIDSIZE | Indicates the trace unit Virtual context identifier size. 0b00100 32-bit Virtual context identifier size. | |
| [9:5] | CIDSIZE | Indicates the Context identifier size. 0b00100 32-bit Context identifier size. | |
| [4:0] | IASIZE | Virtual instruction address size. 0b01000 Maximum of 64-bit instruction address size. | |

Access

MRS <Xt>, TRCIDR2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIDR2 | 0b10 | 0b001 | 0b0000 | 0b1010 | 0b111 |

Accessibility

MRS <Xt>, TRCIDR2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR2;
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return TRCIDR2;
    elsif PSTATE.EL == EL3 then

```

```
if CPTR_EL3.TTA == '1' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    return TRCIDR2;
```

A.9.4 TRCIDR3, ID Register 3

Returns the base architecture of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

See individual bit resets

Bit descriptions

Figure A-108: AArch64_trcidr3 bit assignments

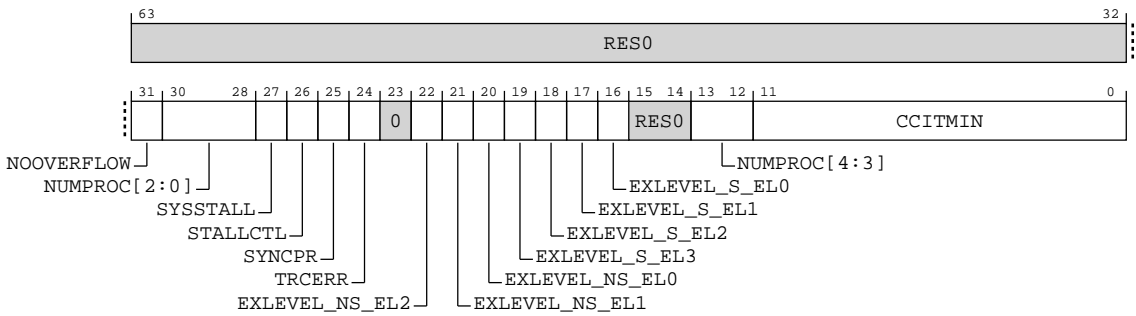


Table A-299: TRCIDR3 bit descriptions

| Bits | Name | Description | Reset |
|----------------|------------|--|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31] | NOOVERFLOW | Indicates if overflow prevention is implemented. 0b0 Overflow prevention is not implemented. | |
| [13:12, 30:28] | NUMPROC | Indicates the number of PEs available for tracing. 0b00000 The trace unit can trace one PE. | |

| Bits | Name | Description | Reset |
|---------|----------------|---|-------|
| [27] | SYSSTALL | Indicates if stalling of the PE is permitted. 0b0 Stalling of the PE is not permitted. | |
| [26] | STALLCTL | Indicates if trace unit implements stalling of the PE. 0b0 Stalling of the PE is not implemented. | |
| [25] | SYNCPR | Indicates if an implementation has a fixed synchronization period. 0b0 AArch64-TRCSYNCPR is read/write so software can change the synchronization period. | |
| [24] | TRCERR | Indicates forced tracing of System Error exceptions is implemented. 0b1 Forced tracing of System Error exceptions is implemented. | |
| [23] | RES0 | Reserved | 0x0 |
| [22] | EXLEVEL_NS_EL2 | Indicates if Non-secure EL2 implemented. 0b1 Non-secure EL2 is implemented. | |
| [21] | EXLEVEL_NS_EL1 | Indicates if Non-secure EL1 implemented. 0b1 Non-secure EL1 is implemented. | |
| [20] | EXLEVEL_NS_ELO | Indicates if Non-secure ELO implemented. 0b1 Non-secure ELO is implemented. | |
| [19] | EXLEVEL_S_EL3 | Indicates if Secure EL3 implemented. 0b1 Secure EL3 is implemented. | |
| [18] | EXLEVEL_S_EL2 | Indicates if Secure EL2 implemented. 0b1 Secure EL2 is implemented. | |
| [17] | EXLEVEL_S_EL1 | Indicates if Secure EL1 implemented. 0b1 Secure EL1 is implemented. | |
| [16] | EXLEVEL_S_ELO | Indicates if Secure ELO implemented. 0b1 Secure ELO is implemented. | |
| [15:14] | RES0 | Reserved | 0b00 |
| [11:0] | CCITMIN | Indicates the minimum value that can be programmed in AArch64-TRCCCCTLR.THRESHOLD. If AArch64-TRCIDR0.TRCCCI == 0b1 then the minimum value of this field is 0x001. If AArch64-TRCIDR0.TRCCCI == 0b0 then this field is zero. 0b000000000100 | |

Access

MRS <Xt>, TRCIDR3

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIDR3 | 0b10 | 0b001 | 0b0000 | 0b1011 | 0b111 |

Accessibility

MRS <Xt>, TRCIDR3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR3;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR3;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR3;

```

A.9.5 TRCIDR4, ID Register 4

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

See individual bit resets

Bit descriptions

Figure A-109: AArch64_trcidr4 bit assignments

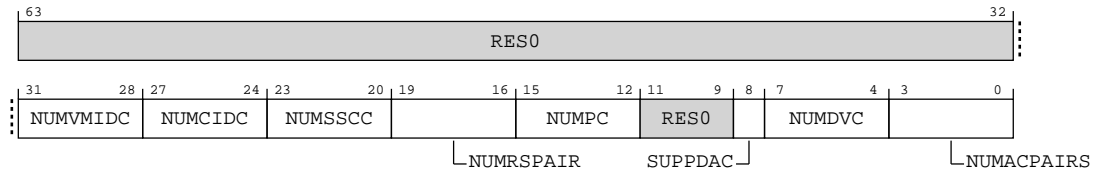


Table A-301: TRCIDR4 bit descriptions

| Bits | Name | Description | Reset |
|---------|------------|--|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:28] | NUMVMIDC | Indicates the number of Virtual Context Identifier Comparators that are available for tracing. 0b0001 The implementation has one Virtual Context Identifier Comparator. | |
| [27:24] | NUMCIDC | Indicates the number of Context Identifier Comparators that are available for tracing. 0b0001 The implementation has one Context Identifier Comparator. | |
| [23:20] | NUMSSCC | Indicates the number of Single-shot Comparator Controls that are available for tracing. 0b0001 The implementation has one Single-shot Comparator Control. | |
| [19:16] | NUMRSPAIR | Indicates the number of resource selector pairs that are available for tracing. 0b0111 The implementation has eight resource selector pairs. | |
| [15:12] | NUMPC | Indicates the number of PE Comparator Inputs that are available for tracing. 0b0000 No PE Comparator Inputs are available. | |
| [11:9] | RES0 | Reserved | 0b000 |
| [8] | SUPPDAC | Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. 0b0 Data address comparisons not implemented. | |
| [7:4] | NUMDVC | Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. 0b0000 No data value comparators implemented. | |
| [3:0] | NUMACPAIRS | Indicates the number of Address Comparator pairs that are available for tracing. 0b0100 The implementation has four Address Comparator pairs. | |

Access

MRS <Xt>, TRCIDR4

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIDR4 | 0b10 | 0b001 | 0b0000 | 0b1100 | 0b111 |

Accessibility

MRS <Xt>, TRCIDR4

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;

```

A.9.6 TRCIDR5, ID Register 5

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

See individual bit resets

Bit descriptions

Figure A-110: AArch64_trcidr5 bit assignments

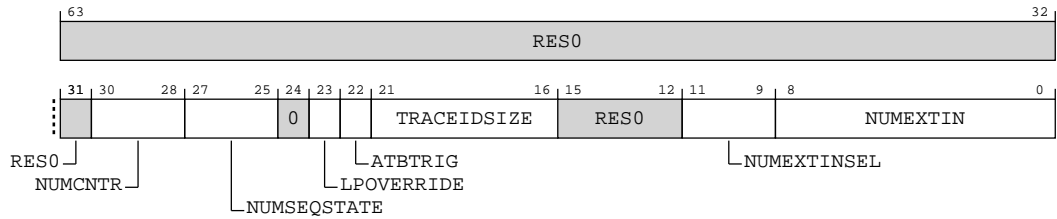


Table A-303: TRCIDR5 bit descriptions

| Bits | Name | Description | Reset |
|---------|-------------|--|--------|
| [63:31] | RES0 | Reserved | 0x0 |
| [30:28] | NUMCNTR | Indicates the number of Counters that are available for tracing. 0b010 Two Counters implemented. | |
| [27:25] | NUMSEQSTATE | Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. 0b100 Four Sequencer states are implemented. | |
| [24] | RES0 | Reserved | 0b0 |
| [23] | LPOVERRIDE | Indicates support for Low-power Override Mode. 0b0 The trace unit does not support Low-power Override Mode. | |
| [22] | ATBTRIG | Indicates if the implementation can support ATB triggers. 0b1 The implementation supports ATB triggers. | |
| [21:16] | TRACEIDSIZE | Indicates the trace ID width. 0b000111 The implementation supports a 7-bit trace ID. | |
| [15:12] | RES0 | Reserved | 0b0000 |
| [11:9] | NUMEXTINSEL | Indicates how many External Input Selector resources are implemented. 0b100 4 External Input Selector resources are available. | |
| [8:0] | NUMEXTIN | Indicates how many External Inputs are implemented. 0b11111111 Unified PMU event selection. | |

Access

MRS <Xt>, TRCIDR5

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIDR5 | 0b10 | 0b001 | 0b0000 | 0b1101 | 0b111 |

Accessibility

MRS <Xt>, TRCIDR5

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;

```

A.9.7 TRCIDR10, ID Register 10

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

0x0

Bit descriptions

Figure A-111: AArch64_trcidr10 bit assignments

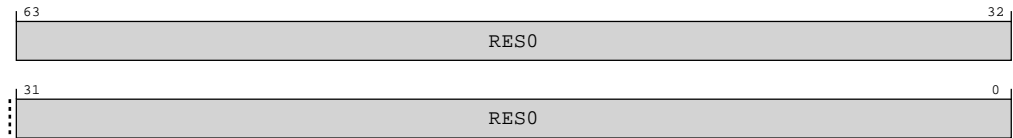


Table A-305: TRCIDR10 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, TRCIDR10

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIDR10 | 0b10 | 0b001 | 0b0000 | 0b0010 | 0b110 |

Accessibility

MRS <Xt>, TRCIDR10

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;

```

A.9.8 TRCIDR11, ID Register 11

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

0x0

Bit descriptions

Figure A-112: AArch64_trcidr11 bit assignments

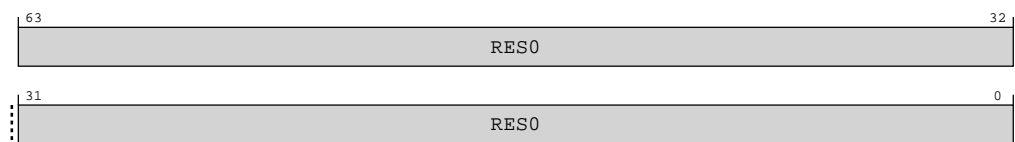


Table A-307: TRCIDR11 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, TRCIDR11

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIDR11 | 0b10 | 0b001 | 0b0000 | 0b0011 | 0b110 |

Accessibility

MRS <Xt>, TRCIDR11

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then

```

```
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR11;
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return TRCIDR11;
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return TRCIDR11;
```

A.9.9 TRCIDR12, ID Register 12

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

0x0

Bit descriptions

Figure A-113: AArch64_trcidr12 bit assignments

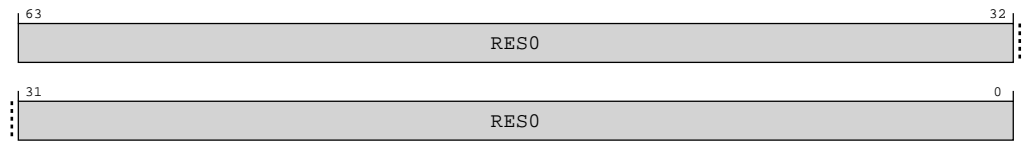


Table A-309: TRCIDR12 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, TRCIDR12

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIDR12 | 0b10 | 0b001 | 0b0000 | 0b0100 | 0b110 |

Accessibility

MRS <Xt>, TRCIDR12

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR12;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR12;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR12;

```

A.9.10 TRCIDR13, ID Register 13

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

0x0

Bit descriptions

Figure A-114: AArch64_trcidr13 bit assignments

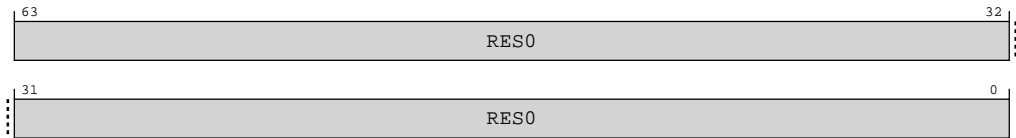


Table A-311: TRCIDR13 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, TRCIDR13

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIDR13 | 0b10 | 0b001 | 0b0000 | 0b0101 | 0b110 |

Accessibility

MRS <Xt>, TRCIDR13

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;

```

A.9.11 TRCIDR0, ID Register 0

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

See individual bit resets

Bit descriptions

Figure A-115: AArch64_trcidr0 bit assignments

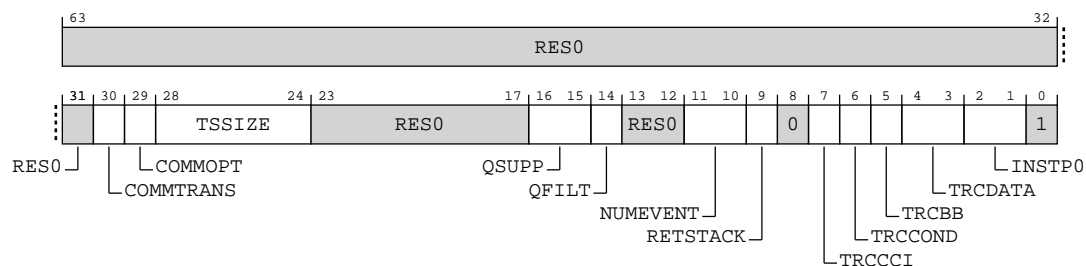


Table A-313: TRCIDR0 bit descriptions

| Bits | Name | Description | Reset |
|---------|-----------|---|-----------|
| [63:31] | RES0 | Reserved | 0x0 |
| [30] | COMMTRANS | Transaction Start element behavior. 0b0 Transaction Start elements are P0 elements. | |
| [29] | COMMOPT | Indicates the contents and encodings of Cycle count packets. 0b1 Commit mode 1. | |
| [28:24] | TSSIZE | Indicates that the trace unit implements Global timestamping and the size of the timestamp value. 0b01000 Global timestamping implemented with a 64-bit timestamp value. | |
| [23:17] | RES0 | Reserved | 0b0000000 |
| [16:15] | QSUPP | Indicates that the trace unit implements Q element support. 0b00 Q element support is not implemented. | |

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [14] | QFILT | Indicates if the trace unit implements Q element filtering. 0b0 Q element filtering is not implemented. | |
| [13:12] | RES0 | Reserved | 0b00 |
| [11:10] | NUMEVENT | Indicates the number of ETEEvents implemented. 0b11 The trace unit supports 4 ETEEvents. | |
| [9] | RETSTACK | Indicates if the trace unit supports the return stack. 0b1 Return stack implemented. | |
| [8] | RES0 | Reserved | 0b0 |
| [7] | TRCCCI | Indicates if the trace unit implements cycle counting. 0b1 Cycle counting implemented. | |
| [6] | TRCCOND | Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. 0b0 Conditional instruction tracing not implemented. | |
| [5] | TRCBB | Indicates if the trace unit implements branch broadcasting. 0b1 Branch broadcasting implemented. | |
| [4:3] | TRCDATA | Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. 0b00 Tracing of data addresses and data values is not implemented. | |
| [2:1] | INSTPO | Indicates if load and store instructions are PO instructions. Load and store instructions as PO instructions is not implemented in ETE and this field is reserved for other trace architectures. 0b00 Load and store instructions are not PO instructions. | |
| [0] | RES1 | Reserved | 0b1 |

Access

MRS <Xt>, TRCIDRO

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIDRO | 0b10 | 0b001 | 0b0000 | 0b1000 | 0b111 |

Accessibility

MRS <Xt>, TRCIDRO

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then

```

```

    AArch64.SystemAccessTrap(EL1, 0x18);
elseif EL2Enabled() && CPTR_EL2.TTA == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elseif CPTR_EL3.TTA == '1' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    return TRCIDR0;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR0;
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR0;

```

A.9.12 TRCIDR1, ID Register 1

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

See individual bit resets

Bit descriptions

Figure A-116: AArch64_trcidr1 bit assignments

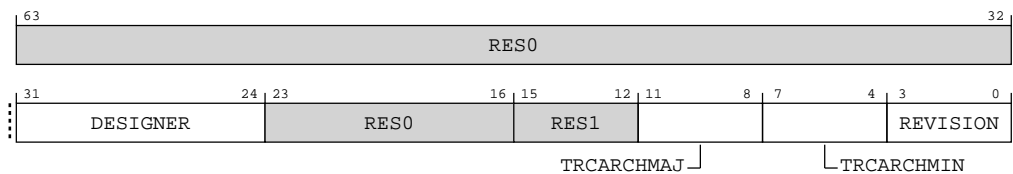


Table A-315: TRCIDR1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|-------------|-------|
| [63:32] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|---------|------------|---|------------|
| [31:24] | DESIGNER | Indicates which company designed the trace unit. The permitted values of this field are the same as AArch64-MIDR_EL1.Implementer. 0b01000001 Arm Limited | |
| [23:16] | RES0 | Reserved | 0b00000000 |
| [15:12] | RES1 | Reserved | 0b1111 |
| [11:8] | TRCARCHMAJ | Major architecture version. 0b1111 If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH. | |
| [7:4] | TRCARCHMIN | Minor architecture version. 0b1111 If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH. | |
| [3:0] | REVISION | Implementation revision that identifies the revision of the trace and OS Lock registers. 0b0000 Revision 0 | |

Access

MRS <Xt>, TRCIDR1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCIDR1 | 0b10 | 0b001 | 0b0000 | 0b1001 | 0b111 |

Accessibility

MRS <Xt>, TRCIDR1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;

```

A.9.13 TRCCIDCVR0, Context Identifier Comparator Value Registers <n>

Contains a Context identifier value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

See individual bit resets

Bit descriptions

Figure A-117: AArch64_trccidcvr0 bit assignments

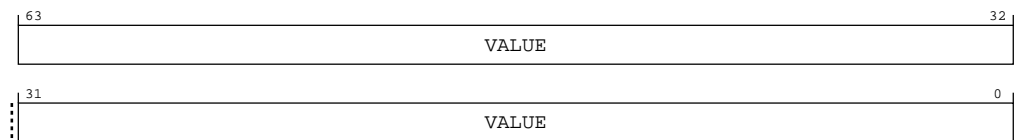


Table A-317: TRCCIDCVR0 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|---|-------|
| [63:0] | VALUE | Context identifier value. The width of this field is indicated by AArch64-TRCIDR2.CIDSIZE. Unimplemented bits are RES0. After a PE Reset, the trace unit assumes that the Context identifier is zero until the PE updates the Context identifier. | |

Access

MRS <Xt>, TRCCIDCVR0

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCCIDCVR0 | 0b10 | 0b001 | 0b0011 | 0b0000 | 0b000 |

MSR TRCCIDCVR0, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| TRCCIDCVR0 | 0b10 | 0b001 | 0b0011 | 0b0000 | 0b000 |

A.10 AArch64 MPAM register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Memory system resource Partitioning And Monitoring (MPAM) registers in the core. For more information about a register, you can click the register name in the table.

Table A-320: AArch64 MPAM register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|------------------------------|-----|-----|-----|-----|-----|---------------------------|--------|---|
| MPAMVPMV_EL2 | 3 | C10 | 4 | C4 | 1 | See individual bit resets | 64-bit | MPAM Virtual Partition Mapping Valid Register |
| MPAMVPM0_EL2 | 3 | C10 | 4 | C6 | 0 | See individual bit resets | 64-bit | MPAM Virtual PARTID Mapping Register 0 |
| MPAMVPM1_EL2 | 3 | C10 | 4 | C6 | 1 | See individual bit resets | 64-bit | MPAM Virtual PARTID Mapping Register 1 |
| MPAMVPM2_EL2 | 3 | C10 | 4 | C6 | 2 | See individual bit resets | 64-bit | MPAM Virtual PARTID Mapping Register 2 |
| MPAMVPM3_EL2 | 3 | C10 | 4 | C6 | 3 | See individual bit resets | 64-bit | MPAM Virtual PARTID Mapping Register 3 |
| MPAMVPM4_EL2 | 3 | C10 | 4 | C6 | 4 | See individual bit resets | 64-bit | MPAM Virtual PARTID Mapping Register 4 |
| MPAMVPM5_EL2 | 3 | C10 | 4 | C6 | 5 | See individual bit resets | 64-bit | MPAM Virtual PARTID Mapping Register 5 |
| MPAMVPM6_EL2 | 3 | C10 | 4 | C6 | 6 | See individual bit resets | 64-bit | MPAM Virtual PARTID Mapping Register 6 |
| MPAMVPM7_EL2 | 3 | C10 | 4 | C6 | 7 | See individual bit resets | 64-bit | MPAM Virtual PARTID Mapping Register 7 |

A.10.1 MPAMVPMV_EL2, MPAM Virtual Partition Mapping Valid Register

Valid bits for virtual PARTID mapping entries. Each bit *m* corresponds to virtual PARTID mapping entry *m* in the MPAMVPM<*n*>_EL2 registers where *n* = *m* >> 2.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

Reset value

See individual bit resets

Bit descriptions

Figure A-118: AArch64_mpmvpmv_el2 bit assignments

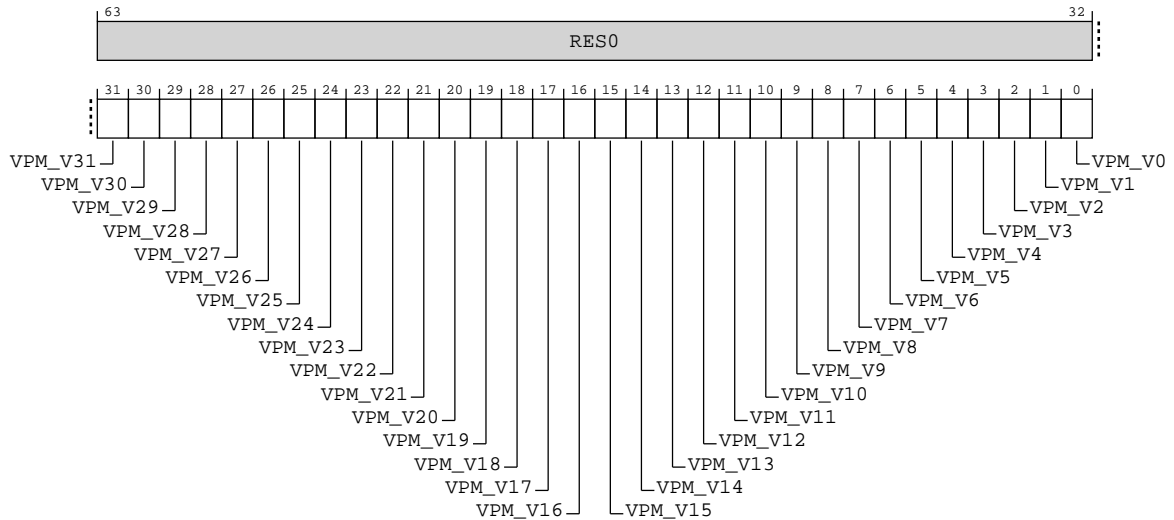


Table A-321: MPAMVPMV_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|---------|---|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31] | VPM_V31 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [30] | VPM_V30 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [29] | VPM_V29 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [28] | VPM_V28 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [27] | VPM_V27 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [26] | VPM_V26 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [25] | VPM_V25 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [24] | VPM_V24 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [23] | VPM_V23 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [22] | VPM_V22 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [21] | VPM_V21 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [20] | VPM_V20 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [19] | VPM_V19 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [18] | VPM_V18 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [17] | VPM_V17 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [16] | VPM_V16 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [15] | VPM_V15 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [14] | VPM_V14 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [13] | VPM_V13 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [12] | VPM_V12 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [11] | VPM_V11 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |

| Bits | Name | Description | Reset |
|------|---------|---|-------|
| [10] | VPM_V10 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [9] | VPM_V9 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [8] | VPM_V8 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [7] | VPM_V7 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [6] | VPM_V6 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [5] | VPM_V5 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [4] | VPM_V4 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [3] | VPM_V3 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [2] | VPM_V2 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [1] | VPM_V1 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |
| [0] | VPM_V0 | Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>. | |

Access

MRS <Xt>, MPAMVPMV_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPMV_EL2 | 0b11 | 0b100 | 0b1010 | 0b0100 | 0b001 |

MSR MPAMVPMV_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPMV_EL2 | 0b11 | 0b100 | 0b1010 | 0b0100 | 0b001 |

Accessibility

MRS <Xt>, MPAMVPMV_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x938];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elsif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPMV_EL2;
    elsif PSTATE.EL == EL3 then
        return MPAMVPMV_EL2;

```

MSR MPAMVPMV_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;

```

```

elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x938] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            MPAMVPMV_EL2 = X[t];
    elseif PSTATE.EL == EL3 then
        MPAMVPMV_EL2 = X[t];

```

A.10.2 MPAMVPMO_EL2, MPAM Virtual PARTID Mapping Register 0

MPAMVPMO_EL2 provides mappings from virtual PARTIDs 0 - 3 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 register. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPMO_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAMO_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

Reset value

See individual bit resets

Bit descriptions

Figure A-119: AArch64_mpamvpm0_el2 bit assignments

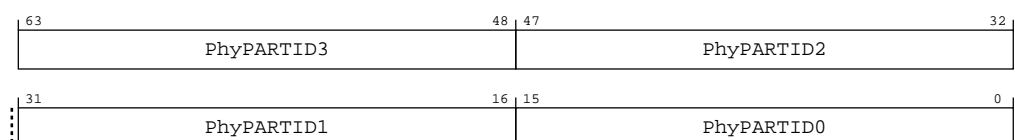


Table A-324: MPAMVPMO_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|------------|---|-------|
| [63:48] | PhyPARTID3 | Virtual PARTID Mapping Entry for virtual PARTID 3. PhyPARTID3 gives the mapping of virtual PARTID 3 to a physical PARTID. | |
| [47:32] | PhyPARTID2 | Virtual PARTID Mapping Entry for virtual PARTID 2. PhyPARTID2 gives the mapping of virtual PARTID 2 to a physical PARTID. | |
| [31:16] | PhyPARTID1 | Virtual PARTID Mapping Entry for virtual PARTID 1. PhyPARTID1 gives the mapping of virtual PARTID 1 to a physical PARTID. | |
| [15:0] | PhyPARTID0 | Virtual PARTID Mapping Entry for virtual PARTID 0. PhyPARTID0 gives the mapping of virtual PARTID 0 to a physical PARTID. | |

Access

MRS <Xt>, MPAMVPMO_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPMO_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b000 |

MSR MPAMVPMO_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPMO_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b000 |

Accessibility

MRS <Xt>, MPAMVPMO_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x940];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPMO_EL2;
    elseif PSTATE.EL == EL3 then
        return MPAMVPMO_EL2;

```

MSR MPAMVPMO_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x940] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then

```

```

        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM0_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    MPAMVPM0_EL2 = X[t];

```

A.10.3 MPAMVPM1_EL2, MPAM Virtual PARTID Mapping Register 1

MPAMVPM1_EL2 provides mappings from virtual PARTIDs 4 - 7 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented AArch64-MPAMVPM0_EL2 to AArch64-MPAMVPM7_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by MPAMHCR_EL2.ELO_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

Reset value

See individual bit resets

Bit descriptions

Figure A-120: AArch64_mpamvpm1_el2 bit assignments

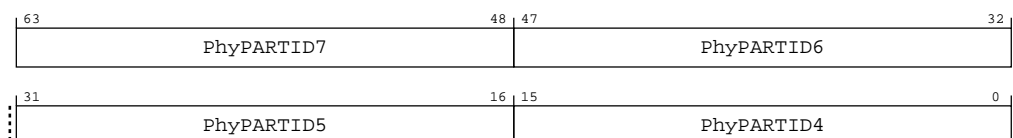


Table A-327: MPAMVPM1_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|------------|---|-------|
| [63:48] | PhyPARTID7 | Virtual PARTID Mapping Entry for virtual PARTID 7. PhyPARTID7 gives the mapping of virtual PARTID 7 to a physical PARTID. | |
| [47:32] | PhyPARTID6 | Virtual PARTID Mapping Entry for virtual PARTID 6. PhyPARTID6 gives the mapping of virtual PARTID 6 to a physical PARTID. | |
| [31:16] | PhyPARTID5 | Virtual PARTID Mapping Entry for virtual PARTID 5. PhyPARTID5 gives the mapping of virtual PARTID 5 to a physical PARTID. | |
| [15:0] | PhyPARTID4 | Virtual PARTID Mapping Entry for virtual PARTID 4. PhyPARTID4 gives the mapping of virtual PARTID 4 to a physical PARTID. | |

Access

MRS <Xt>, MPAMVPM1_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM1_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b001 |

MSR MPAMVPM1_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM1_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b001 |

Accessibility

MRS <Xt>, MPAMVPM1_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x948];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPM1_EL2;
    elseif PSTATE.EL == EL3 then
        return MPAMVPM1_EL2;

```

MSR MPAMVPM1_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x948] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then

```

```

        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM1_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    MPAMVPM1_EL2 = X[t];

```

A.10.4 MPAMVPM2_EL2, MPAM Virtual PARTID Mapping Register 2

MPAMVPM2_EL2 provides mappings from virtual PARTIDs 8 - 11 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented AArch64-MPAMVPM0_EL2 to AArch64-MPAMVPM7_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

Reset value

See individual bit resets

Bit descriptions

Figure A-121: AArch64_mpamvpm2_el2 bit assignments

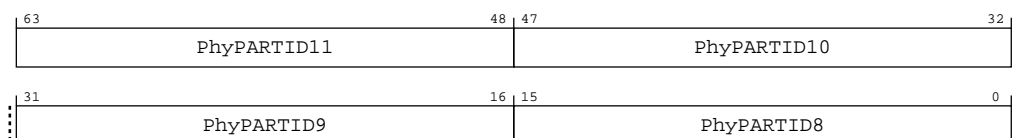


Table A-330: MPAMVPM2_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|-------------|--|-------|
| [63:48] | PhyPARTID11 | Virtual PARTID Mapping Entry for virtual PARTID 11. PhyPARTID11 gives the mapping of virtual PARTID 11 to a physical PARTID. | |
| [47:32] | PhyPARTID10 | Virtual PARTID Mapping Entry for virtual PARTID 10. PhyPARTID10 gives the mapping of virtual PARTID 10 to a physical PARTID. | |
| [31:16] | PhyPARTID9 | Virtual PARTID Mapping Entry for virtual PARTID 9. PhyPARTID9 gives the mapping of virtual PARTID 9 to a physical PARTID. | |
| [15:0] | PhyPARTID8 | Virtual PARTID Mapping Entry for virtual PARTID 8. PhyPARTID8 gives the mapping of virtual PARTID 8 to a physical PARTID. | |

Access

MRS <Xt>, MPAMVPM2_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM2_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b010 |

MSR MPAMVPM2_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM2_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b010 |

Accessibility

MRS <Xt>, MPAMVPM2_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x950];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPM2_EL2;
    elseif PSTATE.EL == EL3 then
        return MPAMVPM2_EL2;

```

MSR MPAMVPM2_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x950] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then

```

```
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM2_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    MPAMVPM2_EL2 = X[t];
```

A.10.5 MPAMVPM3_EL2, MPAM Virtual PARTID Mapping Register 3

MPAMVPM3_EL2 provides mappings from virtual PARTIDs 12 - 15 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.ELO_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

Reset value

See individual bit resets

Bit descriptions

Figure A-122: AArch64_mpamvpm3_el2 bit assignments

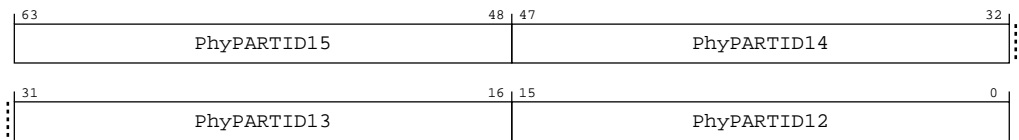


Table A-333: MPAMVPM3_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|-------------|--|-------|
| [63:48] | PhyPARTID15 | Virtual PARTID Mapping Entry for virtual PARTID 15. PhyPARTID15 gives the mapping of virtual PARTID 15 to a physical PARTID. | |
| [47:32] | PhyPARTID14 | Virtual PARTID Mapping Entry for virtual PARTID 14. PhyPARTID14 gives the mapping of virtual PARTID 14 to a physical PARTID. | |
| [31:16] | PhyPARTID13 | Virtual PARTID Mapping Entry for virtual PARTID 13. PhyPARTID13 gives the mapping of virtual PARTID 13 to a physical PARTID. | |
| [15:0] | PhyPARTID12 | Virtual PARTID Mapping Entry for virtual PARTID 12. PhyPARTID12 gives the mapping of virtual PARTID 12 to a physical PARTID. | |

Access

MRS <Xt>, MPAMVPM3_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM3_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b011 |

MSR MPAMVPM3_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM3_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b011 |

Accessibility

MRS <Xt>, MPAMVPM3_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x958];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPM3_EL2;
    elseif PSTATE.EL == EL3 then
        return MPAMVPM3_EL2;

```

MSR MPAMVPM3_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x958] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then

```

```

        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM3_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    MPAMVPM3_EL2 = X[t];

```

A.10.6 MPAMVPM4_EL2, MPAM Virtual PARTID Mapping Register 4

MPAMVPM4_EL2 provides mappings from virtual PARTIDs 16 - 19 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.ELO_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

Reset value

See individual bit resets

Bit descriptions

Figure A-123: AArch64_mpamvpm4_el2 bit assignments

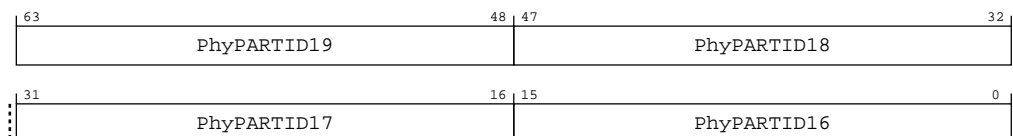


Table A-336: MPAMVPM4_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|-------------|--|-------|
| [63:48] | PhyPARTID19 | Virtual PARTID Mapping Entry for virtual PARTID 19. PhyPARTID19 gives the mapping of virtual PARTID 19 to a physical PARTID. | |
| [47:32] | PhyPARTID18 | Virtual PARTID Mapping Entry for virtual PARTID 18. PhyPARTID18 gives the mapping of virtual PARTID 18 to a physical PARTID. | |
| [31:16] | PhyPARTID17 | Virtual PARTID Mapping Entry for virtual PARTID 17. PhyPARTID17 gives the mapping of virtual PARTID 17 to a physical PARTID. | |
| [15:0] | PhyPARTID16 | Virtual PARTID Mapping Entry for virtual PARTID 16. PhyPARTID16 gives the mapping of virtual PARTID 16 to a physical PARTID. | |

Access

MRS <Xt>, MPAMVPM4_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM4_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b100 |

MSR MPAMVPM4_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM4_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b100 |

Accessibility

MRS <Xt>, MPAMVPM4_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x960];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPM4_EL2;
    elseif PSTATE.EL == EL3 then
        return MPAMVPM4_EL2;

```

MSR MPAMVPM4_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x960] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then

```

```
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM4_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    MPAMVPM4_EL2 = X[t];
```

A.10.7 MPAMVPM5_EL2, MPAM Virtual PARTID Mapping Register 5

MPAMVPM5_EL2 provides mappings from virtual PARTIDs 20 - 23 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.ELO_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

Reset value

See individual bit resets

Bit descriptions

Figure A-124: AArch64_mpamvpm5_el2 bit assignments

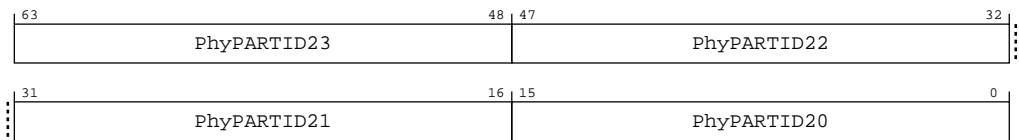


Table A-339: MPAMVPM5_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|-------------|--|-------|
| [63:48] | PhyPARTID23 | Virtual PARTID Mapping Entry for virtual PARTID 23. PhyPARTID23 gives the mapping of virtual PARTID 23 to a physical PARTID. | |
| [47:32] | PhyPARTID22 | Virtual PARTID Mapping Entry for virtual PARTID 22. PhyPARTID22 gives the mapping of virtual PARTID 22 to a physical PARTID. | |
| [31:16] | PhyPARTID21 | Virtual PARTID Mapping Entry for virtual PARTID 21. PhyPARTID21 gives the mapping of virtual PARTID 21 to a physical PARTID. | |
| [15:0] | PhyPARTID20 | Virtual PARTID Mapping Entry for virtual PARTID 20. PhyPARTID20 gives the mapping of virtual PARTID 20 to a physical PARTID. | |

Access

MRS <Xt>, MPAMVPM5_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM5_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b101 |

MSR MPAMVPM5_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM5_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b101 |

Accessibility

MRS <Xt>, MPAMVPM5_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x968];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPM5_EL2;
    elseif PSTATE.EL == EL3 then
        return MPAMVPM5_EL2;

```

MSR MPAMVPM5_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x968] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then

```

```
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM5_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    MPAMVPM5_EL2 = X[t];
```

A.10.8 MPAMVPM6_EL2, MPAM Virtual PARTID Mapping Register 6

MPAMVPM6_EL2 provides mappings from virtual PARTIDs 24 - 27 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.ELO_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

Reset value

See individual bit resets

Bit descriptions

Figure A-125: AArch64_mpamvpm6_el2 bit assignments

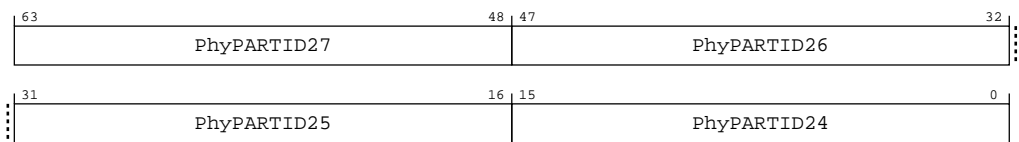


Table A-342: MPAMVPM6_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|-------------|--|-------|
| [63:48] | PhyPARTID27 | Virtual PARTID Mapping Entry for virtual PARTID 27. PhyPARTID27 gives the mapping of virtual PARTID 27 to a physical PARTID. | |
| [47:32] | PhyPARTID26 | Virtual PARTID Mapping Entry for virtual PARTID 26. PhyPARTID26 gives the mapping of virtual PARTID 26 to a physical PARTID. | |
| [31:16] | PhyPARTID25 | Virtual PARTID Mapping Entry for virtual PARTID 25. PhyPARTID25 gives the mapping of virtual PARTID 25 to a physical PARTID. | |
| [15:0] | PhyPARTID24 | Virtual PARTID Mapping Entry for virtual PARTID 24. PhyPARTID24 gives the mapping of virtual PARTID 24 to a physical PARTID. | |

Access

MRS <Xt>, MPAMVPM6_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM6_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b110 |

MSR MPAMVPM6_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM6_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b110 |

Accessibility

MRS <Xt>, MPAMVPM6_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x970];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPM6_EL2;
    elseif PSTATE.EL == EL3 then
        return MPAMVPM6_EL2;

```

MSR MPAMVPM6_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x970] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then

```

```
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM6_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    MPAMVPM6_EL2 = X[t];
```

A.10.9 MPAMVPM7_EL2, MPAM Virtual PARTID Mapping Register 7

MPAMVPM7_EL2 provides mappings from virtual PARTIDs 28 - 31 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.ELO_VPMEN for AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

Reset value

See individual bit resets

Bit descriptions

Figure A-126: AArch64_mpamvpm7_el2 bit assignments

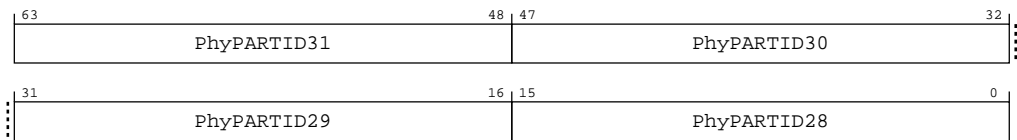


Table A-345: MPAMVPM7_EL2 bit descriptions

| Bits | Name | Description | Reset |
|---------|-------------|--|-------|
| [63:48] | PhyPARTID31 | Virtual PARTID Mapping Entry for virtual PARTID 31. PhyPARTID31 gives the mapping of virtual PARTID 31 to a physical PARTID. | |
| [47:32] | PhyPARTID30 | Virtual PARTID Mapping Entry for virtual PARTID 30. PhyPARTID30 gives the mapping of virtual PARTID 30 to a physical PARTID. | |
| [31:16] | PhyPARTID29 | Virtual PARTID Mapping Entry for virtual PARTID 29. PhyPARTID29 gives the mapping of virtual PARTID 29 to a physical PARTID. | |
| [15:0] | PhyPARTID28 | Virtual PARTID Mapping Entry for virtual PARTID 28. PhyPARTID28 gives the mapping of virtual PARTID 28 to a physical PARTID. | |

Access

MRS <Xt>, MPAMVPM7_EL2

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM7_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b111 |

MSR MPAMVPM7_EL2, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| MPAMVPM7_EL2 | 0b11 | 0b100 | 0b1010 | 0b0110 | 0b111 |

Accessibility

MRS <Xt>, MPAMVPM7_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x978];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPM7_EL2;
    elseif PSTATE.EL == EL3 then
        return MPAMVPM7_EL2;

```

MSR MPAMVPM7_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x978] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then

```

```

        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM7_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    MPAMVPM7_EL2 = X[t];

```

A.11 AArch64 RAS register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** RAS registers in the core. For more information about a register, you can click the register name in the table.

Table A-348: AArch64 RAS register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-------------------------------|-----|-----|-----|-----|-----|---------------------------|--------|---|
| ERRIDR_EL1 | 3 | C5 | 0 | C3 | 0 | See individual bit resets | 64-bit | Error Record ID Register |
| ERRSELR_EL1 | 3 | C5 | 0 | C3 | 1 | See individual bit resets | 64-bit | Error Record Select Register |
| ERXFR_EL1 | 3 | C5 | 0 | C4 | 0 | See individual bit resets | 64-bit | Selected Error Record Feature Register |
| ERXCTLR_EL1 | 3 | C5 | 0 | C4 | 1 | 0x0 | 64-bit | Selected Error Record Control Register |
| ERXSTATUS_EL1 | 3 | C5 | 0 | C4 | 2 | 0x0 | 64-bit | Selected Error Record Primary Status Register |
| ERXADDR_EL1 | 3 | C5 | 0 | C4 | 3 | See individual bit resets | 64-bit | Selected Error Record Address Register |
| ERXPFGF_EL1 | 3 | C5 | 0 | C4 | 4 | See individual bit resets | 64-bit | Selected Pseudo-fault Generation Feature register |
| ERXPFGCTL_EL1 | 3 | C5 | 0 | C4 | 5 | 0x1000 | 64-bit | Selected Pseudo-fault Generation Control register |
| ERXPFGCDN_EL1 | 3 | C5 | 0 | C4 | 6 | See individual bit resets | 64-bit | Selected Pseudo-fault Generation Countdown register |
| ERXMISCO_EL1 | 3 | C5 | 0 | C5 | 0 | See individual bit resets | 64-bit | Selected Error Record Miscellaneous Register 0 |
| ERXMISC1_EL1 | 3 | C5 | 0 | C5 | 1 | 0x0 | 64-bit | Selected Error Record Miscellaneous Register 1 |
| ERXMISC2_EL1 | 3 | C5 | 0 | C5 | 2 | 0x0 | 64-bit | Selected Error Record Miscellaneous Register 2 |
| ERXMISC3_EL1 | 3 | C5 | 0 | C5 | 3 | 0x0 | 64-bit | Selected Error Record Miscellaneous Register 3 |

A.11.1 ERRIDR_EL1, Error Record ID Register

Defines the highest numbered index of the error records that can be accessed through the Error Record System registers.

Configurations

This register is available in all configurations.

Attributes

Width

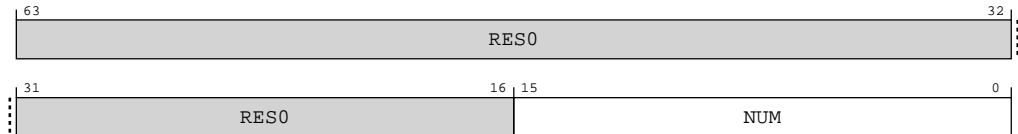
64

Functional group

RAS

Reset value

See individual bit resets

Bit descriptions**Figure A-127: AArch64_erridr_el1 bit assignments****Table A-349: ERRIDR_EL1 bit descriptions**

| Bits | Name | Description | Reset |
|---------|------|--|-------|
| [63:16] | RES0 | Reserved | 0x0 |
| [15:0] | NUM | <p>Highest numbered index of the records that can be accessed through the Error Record System registers plus one. Zero indicates no records can be accessed through the Error Record System registers.</p> <p>Each implemented record is owned by a node. A node might own multiple records.</p> <p>0b0000000000000001</p> <p>One Record Present.</p> | |

Access

MRS <Xt>, ERRIDR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ERRIDR_EL1 | 0b11 | 0b000 | 0b0101 | 0b0011 | 0b000 |

Accessibility

MRS <Xt>, ERRIDR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERRIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRIDR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRIDR_EL1;
elsif PSTATE.EL == EL3 then

```

```
return ERRIDR_EL1;
```

A.11.2 ERRSELR_EL1, Error Record Select Register

Selects an error record to be accessed through the Error Record System registers.

Configurations

If AArch64-ERRIDR_EL1 indicates that zero error records are implemented, then it is **IMPLEMENTATION DEFINED** whether ERRSELR_EL1 is UNDEFINED or RES0.

Attributes

Width

64

Functional group

RAS

Reset value

See individual bit resets

Bit descriptions

Figure A-128: AArch64_errselr_el1 bit assignments

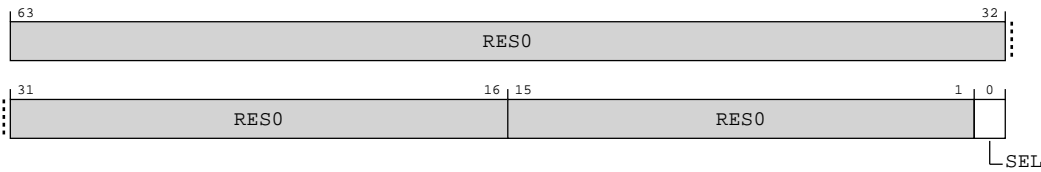


Table A-351: ERRSELR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|--|-------|
| [63:1] | RES0 | Reserved | 0x0 |
| [0] | SEL | 0b0 Selects record 0, containing errors from core RAMs | |

Access

MRS <Xt>, ERRSELR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ERRSELR_EL1 | 0b11 | 0b000 | 0b0101 | 0b0011 | 0b001 |

MSR ERRSELR_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ERRSELR_EL1 | 0b11 | 0b000 | 0b0101 | 0b0011 | 0b001 |

Accessibility

MRS <Xt>, ERRSELR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERRSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRSELR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRSELR_EL1;
elsif PSTATE.EL == EL3 then
    return ERRSELR_EL1;

```

MSR ERRSELR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERRSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERRSELR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERRSELR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERRSELR_EL1 = X[t];

```

A.11.3 ERXFR_EL1, Selected Error Record Feature Register

Accesses ext-ERR<n>FR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

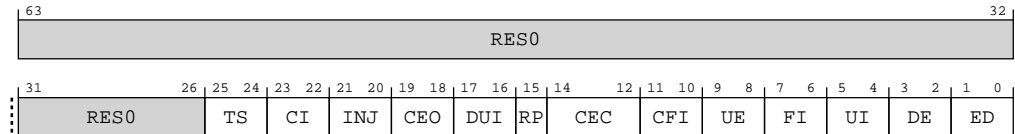
64

Functional group

RAS

Reset value

See individual bit resets

Bit descriptions**Figure A-129: AArch64_erxfr_el1 bit assignments****Table A-354: ERXFR_EL1 bit descriptions**

| Bits | Name | Description | Reset |
|---------|------|--|----------|
| [63:26] | RES0 | Reserved | 0b000000 |
| [25:24] | TS | Timestamp Extension. Indicates whether, for each error record <m> owned by this node, rERXMISC3_EL1 is used as the timestamp register, and, if it is, the timebase used by the timestamp. 0b00 The node does not support a timestamp register. | |
| [23:22] | CI | Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented. 0b00 Does not support the critical error interrupt. ERXCTLR_EL1.CI is RES0. | |
| [21:20] | INJ | Fault Injection Extension. Indicates whether the RAS Common Fault Injection Model Extension is implemented. 0b01 The node implements the RAS Common Fault Injection Model Extension. See ERXPFGF_EL1 for more information. | |
| [19:18] | CEO | Corrected Error overwrite. Indicates the behavior when a second Corrected error is detected after a first Corrected error has been recorded by an error record <m> owned by the node. 0b00 Counts Corrected errors if a counter is implemented. Keeps the previous error syndrome. If the counter overflows, or no counter is implemented, then ERXSTATUS_EL1.OF is set to 0b1. | |
| [17:16] | DUI | Error recovery interrupt for deferred errors control. Indicates whether the control for enabling error recovery interrupts on deferred errors are implemented. 0b00 Does not support the control for enabling error recovery interrupts on deferred errors. ERXCTLR_EL1.DUI is RES0. | |
| [15] | RP | Repeat counter. Indicates whether the node implements the repeat Corrected error counter in ERXMISC0_EL1 for each error record <m> owned by the node that implements the standard Corrected error counter. 0b1 A first (repeat) counter and a second (other) counter are implemented. The repeat counter is the same size as the primary error counter. | |

| Bits | Name | Description | Reset |
|---------|------|--|-------|
| [14:12] | CEC | Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter (CE counter) mechanisms in ERXMISCO_EL1 for each error record <m> owned by the node that can record countable errors. 0b010 Implements an 8-bit Corrected error counter in ERXMISCO_EL1[39:32]. | |
| [11:10] | CFI | Fault handling interrupt for corrected errors. Indicates whether the control for enabling fault handling interrupts on corrected errors are implemented. 0b10 Control for enabling fault handling interrupts on corrected errors is supported and controllable using ERXCTLR_EL1.CFI. | |
| [9:8] | UE | In-band uncorrected error reporting. Indicates whether the in-band uncorrected error reporting (External Aborts) and associated controls are implemented. 0b01 In-band uncorrected error reporting (External Aborts) is supported and always enabled. ERXCTLR_EL1.UE is RES0. | |
| [7:6] | FI | Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented. 0b10 Fault handling interrupt is supported and controllable using ERXCTLR_EL1.FI. | |
| [5:4] | UI | Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented. 0b10 Error handling interrupt is supported and controllable using ERXCTLR_EL1.UI. | |
| [3:2] | DE | 0b00 | |
| [1:0] | ED | Error reporting and logging. Indicates whether error record <n> is the first record owned the node, and, if so, whether it implements the controls for enabling and disabling error reporting and logging. 0b10 Error reporting and logging is controllable using ERXCTLR_EL1.ED. | |

Access

MRS <Xt>, ERXFR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ERXFR_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b000 |

Accessibility

MRS <Xt>, ERXFR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else

```

```
        return ERXFR_EL1;
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.TERR == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return ERXFR_EL1;
    elsif PSTATE.EL == EL3 then
        return ERXFR_EL1;
```

A.11.4 ERXCTLR_EL1, Selected Error Record Control Register

Accesses ext-ERR<n>CTLR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

0x0

Bit descriptions

Figure A-130: AArch64_erxctlr_el1 bit assignments

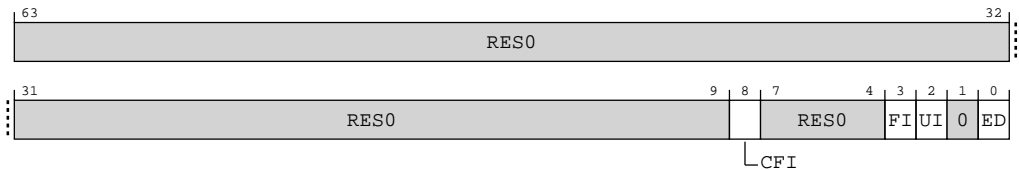


Table A-356: ERXCTLR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:9] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|------|---|--------|
| [8] | CFI | <p>Fault handling interrupt for Corrected errors enable.</p> <p>This control applies to errors arising from both reads and writes.</p> <p>The fault handling interrupt is generated when one of the standard CE counters on ERXMISC0_EL1 overflows and the overflow bit is set. The possible values are:</p> <p>0b0</p> <p>Fault handling interrupt not generated for Corrected errors.</p> <p>0b1</p> <p>Fault handling interrupt generated for Corrected errors.</p> <p>The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0x0 |
| [7:4] | RES0 | Reserved | 0b0000 |
| [3] | FI | <p>Fault handling interrupt enable.</p> <p>This control applies to errors arising from both reads and writes.</p> <p>The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. The possible values are:</p> <p>0b0</p> <p>Fault handling interrupt disabled.</p> <p>0b1</p> <p>Fault handling interrupt enabled.</p> <p>The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0b0 |
| [2] | UI | <p>Uncorrected error recovery interrupt enable.</p> <p>This control applies to errors arising from both reads and writes.</p> <p>When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred.</p> <p>0b0</p> <p>Error recovery interrupt disabled.</p> <p>0b1</p> <p>Error recovery interrupt enabled.</p> <p>The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0b0 |
| [1] | RES0 | Reserved | 0b0 |

| Bits | Name | Description | Reset |
|------|------|---|-------|
| [0] | ED | Error Detection and correction enable. The possible values are: 0b0 Error detection and correction disabled. 0b1 Error detection and correction enabled. Cold reset only. Unaffected by Warm reset | 0b0 |

Access

MRS <Xt>, ERXCTLR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ERXCTLR_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b001 |

MSR ERXCTLR_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ERXCTLR_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b001 |

Accessibility

MRS <Xt>, ERXCTLR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXCTLR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXCTLR_EL1;
elsif PSTATE.EL == EL3 then
    return ERXCTLR_EL1;

```

MSR ERXCTLR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXCTLR_EL1 = X[t];

```

```

elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXCTLR_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXCTLR_EL1 = X[t];

```

A.11.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register

Accesses ext-ERR<n>STATUS for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

0x0

Bit descriptions

Figure A-131: AArch64_erxstatus_el1 bit assignments

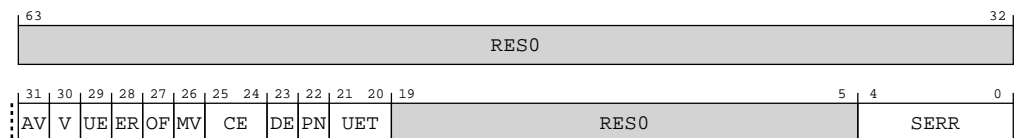


Table A-359: ERXSTATUS_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|--|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31] | AV | <p>Address Valid. The possible values are:</p> <p>0b0</p> <p>ERXADDR_EL1 not valid.</p> <p>0b1</p> <p>ERXADDR_EL1 contains an address associated with the highest priority error recorded by this record.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0x0 |

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [30] | V | <p>Status Register Valid. The possible values are:</p> <p>0b0 ERXSTATUS_EL1 not valid.</p> <p>0b1 ERXSTATUS_EL1 valid. At least one error has been recorded.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0x0 |
| [29] | UE | <p>Uncorrected Error. The possible values are:</p> <p>0b0 No errors have been detected, or all detected errors have been either corrected or deferred.</p> <p>0b1 At least one detected error was not corrected and not deferred.</p> <p>When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero.</p> <p>This bit is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0b0.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0x0 |
| [28] | ER | <p>Error Reported. The possible values are:</p> <p>0b0 No in-band error (External Abort) reported.</p> <p>0b1 An External Abort was signaled by the node to the master making the access or other transaction.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p> <p>Note: An External Abort signaled by the node might be masked and not generate any exception.</p> | 0x0 |

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [27] | OF | <p>Overflow. The possible values are:</p> <p>0b0</p> <p>If UE == 1, then no error status for an Uncorrected error has been discarded.</p> <p>If UE == 0 and DE == 1, then no error status for a Deferred error has been discarded.</p> <p>If UE == 0, DE == 0, and CE != 0b00, then the corrected error counter has not overflowed.</p> <p>0b1</p> <p>More than one error has occurred and so details of the other error have been discarded.</p> <p>When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero.</p> <p>This bit is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0b0.</p> <p>Cold reset only. Unaffected by Warm reset</p> <p>This bit is read/write-one-to-clear.</p> | 0x0 |
| [26] | MV | <p>Miscellaneous Registers Valid. The possible values are:</p> <p>0b0</p> <p>ERXMISC<m>_EL1 not valid.</p> <p>0b1</p> <p>This bit indicates that the ERXMISC<m>_EL1 registers contain additional information for an error recorded by this record.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p> <p>Note:</p> <p>If the ERXMISC<m>_EL1 registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded.</p> | 0x0 |

| Bits | Name | Description | Reset |
|---------|------|--|-------|
| [25:24] | CE | <p>Corrected Error. The possible values are:</p> <p>0b00 No errors were corrected.</p> <p>0b01 At least one transient error was corrected.</p> <p>0b10 At least one error was corrected.</p> <p>0b11 At least one persistent error was corrected.</p> <p>When clearing ERXSTATUS_EL1.V to 0b0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero.</p> <p>This field is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0b0.</p> <p>This field is read/write-ones-to-clear. Writing a value other than all-zeros or all-ones sets this field to an UNKNOWN value.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0x0 |
| [23] | DE | <p>Deferred Error. The possible values are:</p> <p>0b0 No errors were deferred.</p> <p>0b1 At least one error was not corrected and deferred.</p> <p>When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero.</p> <p>This bit is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0b0.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0x0 |
| [22] | PN | <p>Poison. The value is:</p> <p>0b0 This core cannot distinguish a poisoned value from a corrupted value.</p> <p>When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero.</p> <p>This bit is not valid and reads UNKNOWN if any of the following are true:</p> <ul style="list-style-type: none"> ERXSTATUS_EL1.V == 0b0. ERXSTATUS_EL1.{DE,UE} == {0,0}. <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0x0 |

| Bits | Name | Description | Reset |
|---------|------|---|-------|
| [21:20] | UET | Uncorrected Error Type. The value is: 0b00 Uncorrected error, Uncontainable error (UC). Cold reset only. Unaffected by Warm reset | 0x0 |
| [19:5] | RES0 | Reserved | 0x0 |
| [4:0] | SERR | Primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry. The possible values are: 0b00000 No error 0b00010 ECC error from internal data buffer. 0b00110 ECC error on cache data RAM. 0b00111 ECC error on cache tag or dirty RAM. 0b01000 Parity error on TLB data RAM. 0b10010 Error response for a cache copyback. 0b10101 Deferred error from slave not supported at the consumer. For example, poisoned data received from a slave by a master that cannot defer the error further. Cold reset only. Unaffected by Warm reset | 0x0 |

Access

MRS <Xt>, ERXSTATUS_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ERXSTATUS_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b010 |

MSR ERXSTATUS_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ERXSTATUS_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b010 |

Accessibility

MRS <Xt>, ERXSTATUS_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;

```

```

elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXSTATUS_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXSTATUS_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXSTATUS_EL1;
elseif PSTATE.EL == EL3 then
    return ERXSTATUS_EL1;

```

MSR ERXSTATUS_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXSTATUS_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXSTATUS_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXSTATUS_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXSTATUS_EL1 = X[t];

```

A.11.6 ERXADDR_EL1, Selected Error Record Address Register

Accesses ext-ERR<n>ADDR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

See individual bit resets

Bit descriptions

Figure A-132: AArch64_erxaddr_el1 bit assignments

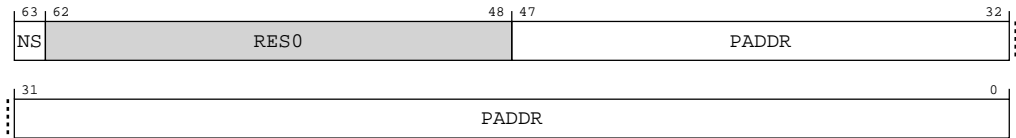


Table A-362: ERXADDR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|-------|--|-------|
| [63] | NS | Non-secure attribute. 0b0 The address is Secure. 0b1 The address is Non-secure. Unaffected by Cold or Warm reset. | |
| [62:48] | RES0 | Reserved | 0x0 |
| [47:0] | PADDR | Physical Address. Address of the recorded location. Unaffected by Cold or Warm reset. | |

Access

MRS <Xt>, ERXADDR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ERXADDR_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b011 |

MSR ERXADDR_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ERXADDR_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b011 |

Accessibility

MRS <Xt>, ERXADDR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXADDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXADDR_EL1;
    elsif PSTATE.EL == EL2 then

```

```

    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXADDR_EL1;
    elsif PSTATE.EL == EL3 then
        return ERXADDR_EL1;

```

MSR ERXADDR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXADDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXADDR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXADDR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXADDR_EL1 = X[t];

```

A.11.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register

Accesses ext-ERR<n>PFGF for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

See individual bit resets

Bit descriptions

Figure A-133: AArch64_erxpfgf_el1 bit assignments

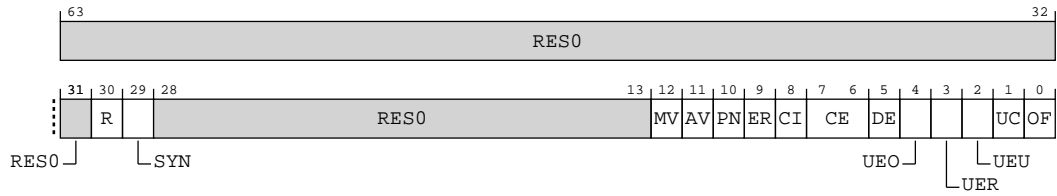


Table A-365: ERXPFGF_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|--|-------|
| [63:31] | RES0 | Reserved | 0x0 |
| [30] | R | Restartable bit. When it reaches zero, the Error Generation Counter restarts from the ERXPFGCDN_EL1 value or stops. The value is: 0b1 Feature controllable. | |
| [29] | SYN | Syndrome. Fault syndrome injection. The value is: 0b0 When an injected error is recorded, the node sets ERXSTATUS_EL1.{IERR, SERR} to IMPLEMENTATION DEFINED values. ERXSTATUS_EL1.{IERR, SERR} are UNKNOWN when ERXSTATUS_EL1.V == 0b0. | |
| [28:13] | RES0 | Reserved | 0x0 |
| [12] | MV | Miscellaneous syndrome. Additional syndrome injection. Defines whether software can control all or part of the syndrome recorded in the ERXMISC<m>_EL1 registers when an injected error is recorded. It is IMPLEMENTATION DEFINED which syndrome fields in ERXMISC<m>_EL1 this refers to, as some fields might always be recorded by an error. For example, a Corrected Error counter. 0b0 When an injected error is recorded, the node might record IMPLEMENTATION DEFINED additional syndrome in ERXMISC<m>_EL1. If any syndrome is recorded in ERXMISC<m>_EL1, then ERXSTATUS_EL1.MV is set to 0b1. Note: If ERR<n>PFGF.MV == 0b1, software can write specific values into the ERR<n>MISC<m> registers when setting up a fault injection event. The values that can be written to these registers are IMPLEMENTATION DEFINED . | |
| [11] | AV | Address syndrome. Address syndrome injection. The value is: 0b0 When an injected error is recorded, the node either sets ERXADDR_EL1 and ERXSTATUS_EL1.AV for the access, or leaves these unchanged. | |
| [10] | PN | Poison flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.PN status flag. The value is: 0b0 When an injected error is recorded, the node sets ERXSTATUS_EL1.PN to 0. | |

| Bits | Name | Description | Reset |
|-------|------|--|-------|
| [9] | ER | Error Reported flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.ER status flag. The value is: 0b0 When an injected error is recorded, the node sets ERXSTATUS_EL1.ER according to the architecture-defined rules for setting the ER bit. | |
| [8] | CI | Critical Error flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.CI status flag. The value is: 0b0 The node does not support this type of flag This behavior replaces the architecture-defined rules for setting the CI bit. | |
| [7:6] | CE | Corrected Error generation. The value is: 0b01 The fault generation feature of the node allows generation of a non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10. All other values are reserved. | |
| [5] | DE | Deferred Error generation. The value is: 0b1 The fault generation feature of the node allows generation of this type of error. | |
| [4] | UEO | Latent or Restartable Error generation. The value is: 0b0 The fault generation feature of the node cannot generate this type of error. | |
| [3] | UER | Signaled or Recoverable Error generation. The value is: 0b0 The fault generation feature of the node cannot generate this type of error. | |
| [2] | UEU | Unrecoverable Error generation. The value is: 0b0 The fault generation feature of the node cannot generate this type of error. | |
| [1] | UC | Uncontainable Error generation. The value is: 0b1 The fault generation feature of the node allows generation of this type of error. | |
| [0] | OF | Overflow flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.OF status flag. The value is: 0b0 When an injected error is recorded, the node sets ERXSTATUS_EL1.OF according to the architecture-defined rules for setting the OF bit. | |

Access

MRS <Xt>, ERXPFGF_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| ERXPFGF_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b100 |

Accessibility

MRS <Xt>, ERXPFGF_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXPFGF_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGF_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGF_EL1;
elseif PSTATE.EL == EL3 then
    return ERXPFGF_EL1;

```

A.11.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register

Accesses ext-ERR<n>PFGCTL for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

0x1000

Bit descriptions

Figure A-134: AArch64_erxpfctl_el1 bit assignments

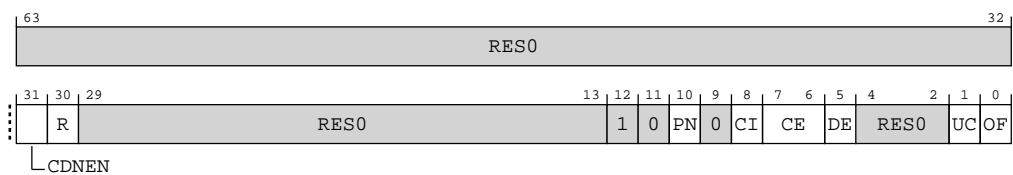


Table A-367: ERXPFGCTL_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|-------|--|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31] | CDNEN | <p>Countdown Enable. Controls transfers from the value that is held in the ERXPFGCDN_EL1 into the Error Generation Counter and enables this counter.</p> <p>0b0</p> <p>The Error Generation Counter is disabled.</p> <p>0b1</p> <p>The Error Generation Counter is enabled. On a write of 0b1 to this bit, the Error Generation Counter is set to ERXPFGCDN_EL1.CDN.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0x0 |
| [30] | R | <p>Restart. Controls whether, upon reaching zero, the Error Generation Counter restarts from the ERXPFGCDN_EL1 value or stops.</p> <p>0b0</p> <p>On reaching 0, the Error Generation Counter will stop.</p> <p>0b1</p> <p>On reaching 0, the Error Generation Counter is set to ERXPFGCDN_EL1.CDN.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0x0 |
| [29:13] | RES0 | Reserved | 0x0 |
| [12] | RES1 | Reserved | 0x1 |
| [11] | RES0 | Reserved | 0x0 |
| [10] | PN | <p>Poison flag. The value that is written to AArch64-ERXSTATUS.PN when an injected error is recorded.</p> <p>0b00</p> <p>AArch64-ERXSTATUS.PN is set to 0 when an injected error is recorded.</p> <p>0b01</p> <p>AArch64-ERXSTATUS.PN is set to 1 when an injected error is recorded.</p> | 0x0 |
| [9] | RES0 | Reserved | 0x0 |
| [8] | CI | <p>Critical Error flag. The value that is written to AArch64-ERXSTATUS.CI when an injected error is recorded.</p> <p>0b00</p> <p>AArch64-ERXSTATUS.CI is set to 0 when an injected error is recorded.</p> <p>0b01</p> <p>AArch64-ERXSTATUS.CI is set to 1 when an injected error is recorded.</p> | 0x0 |
| [7:6] | CE | <p>Corrected Error generation enable. Controls the type of Corrected Error condition that might be generated. The possible values are:</p> <p>0b00</p> <p>No error of this type will be generated.</p> <p>0b01</p> <p>A non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10, might be generated when the Error Generation Counter decrements to zero.</p> <p>Cold reset only. Unaffected by Warm reset</p> | 0x0 |

| Bits | Name | Description | Reset |
|-------|------|--|-------|
| [5] | DE | Deferred Error generation enable. The possible values are: 0b0 No error of this type will be generated. 0b1 An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset | 0x0 |
| [4:2] | RES0 | Reserved | 0b000 |
| [1] | UC | Uncontainable Error generation enable. The possible values are: 0b0 No error of this type will be generated. 0b1 An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset | 0b0 |
| [0] | OF | Overflow flag. The possible values are: 0b0 ERXSTATUS_EL1.OF is set to 0 when an injected error is recorded. 0b1 ERXSTATUS_EL1.OF is set to 1 when an injected error is recorded. Cold reset only. Unaffected by Warm reset | 0b0 |

Access

MRS <Xt>, ERXPFGCTL_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ERXPFGCTL_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b101 |

MSR ERXPFGCTL_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ERXPFGCTL_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b101 |

Accessibility

MRS <Xt>, ERXPFGCTL_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXPFGCTL_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else

```

```

        return ERXPFPGCTL_EL1;
    elseif PSTATE.EL == EL2 then
        if SCR_EL3.FIEN == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return ERXPFPGCTL_EL1;
    elseif PSTATE.EL == EL3 then
        return ERXPFPGCTL_EL1;

```

MSR ERXPFPGCTL_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXPFPGCTL_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFPGCTL_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFPGCTL_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXPFPGCTL_EL1 = X[t];

```

A.11.9 ERXPFPGCDN_EL1, Selected Pseudo-fault Generation Countdown register

Accesses ext-ERR<n>PFGCDN for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

See individual bit resets

Bit descriptions

Figure A-135: AArch64_erxpfgcdn_el1 bit assignments

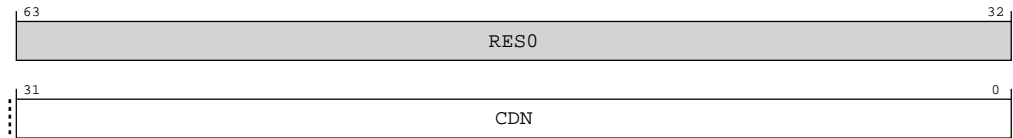


Table A-370: ERXPFGCDN_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|---|-------|
| [63:32] | RES0 | Reserved | 0x0 |
| [31:0] | CDN | <p>Countdown value.</p> <p>This field is copied to Error Generation Counter when either:</p> <ul style="list-style-type: none"> Software writes ERXPFGCTL_EL1.CDNEN with 1. The Error Generation Counter decrements to zero and ERXPFGCTL_EL1.R == 0b1. <p>Unaffected by Cold or Warm reset.</p> <p>Note: The current Error Generation Counter value is not visible to software.</p> | |

Access

MRS <Xt>, ERXPFGCDN_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ERXPFGCDN_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b110 |

MSR ERXPFGCDN_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|---------------|------|-------|--------|--------|-------|
| ERXPFGCDN_EL1 | 0b11 | 0b000 | 0b0101 | 0b0100 | 0b110 |

Accessibility

MRS <Xt>, ERXPFGCDN_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEN == '1' && HFGTR_EL2.ERXPFGCDN_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGCDN_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then

```

```

        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGCDN_EL1;
    elsif PSTATE.EL == EL3 then
        return ERXPFGCDN_EL1;

```

MSR ERXPFGCDN_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXPFGCDN_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXPFGCDN_EL1 = X[t];

```

A.11.10 ERXMISCO_EL1, Selected Error Record Miscellaneous Register 0

Accesses ext-ERR<n>MISCO for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

See individual bit resets

Bit descriptions

Figure A-136: AArch64_erxmisc0_el1 bit assignments

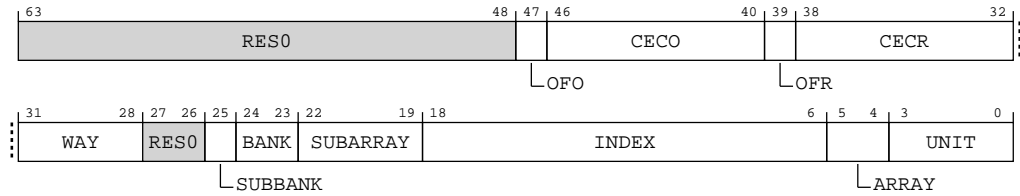


Table A-373: ERXMISC0_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------|---|-------|
| [63:48] | RES0 | Reserved | 0x0 |
| [47] | OFO | <p>Sticky overflow bit, other. Set to 1 when ERXMISC0_EL1.CECO is incremented and wraps through zero.</p> <p>0b0</p> <p>Other counter has not overflowed.</p> <p>0b1</p> <p>Other counter has overflowed.</p> <p>A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an UNKNOWN value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an UNKNOWN value.</p> <p>Unaffected by Cold or Warm reset.</p> | |
| [46:40] | CECO | <p>Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERXMISC0_EL1.CECR.</p> <p>Unaffected by Cold or Warm reset.</p> | |
| [39] | OFR | <p>Sticky overflow bit, repeat. Set to 1 when ERXMISC0_EL1.CECR is incremented and wraps through zero.</p> <p>0b0</p> <p>Repeat counter has not overflowed.</p> <p>0b1</p> <p>Repeat counter has overflowed.</p> <p>A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an UNKNOWN value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an UNKNOWN value.</p> <p>Unaffected by Cold or Warm reset.</p> | |
| [38:32] | CECR | <p>Corrected error count, repeat. Incremented for the first countable error, which also records other syndrome for the error, and subsequently for each countable error that matches the recorded other syndrome.</p> <p>This field resets to an 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.</p> <p>Unaffected by Cold or Warm reset.</p> | |

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [31:28] | WAY | <p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L1 Data Cache]</p> <ul style="list-style-type: none"> Indicates which Tag RAM way or data RAM way detected the error. Upper 2 bits are unused. <p>[L2 TLB]</p> <ul style="list-style-type: none"> Indicates which RAM detected an error. The possible values are 0 (RAM 1) to 9 (RAM 10). <p>[L1 Instruction Cache]</p> <ul style="list-style-type: none"> Indicates which way detected the error. Upper 2 bits are unused. <p>[L2 Cache]</p> <ul style="list-style-type: none"> Indicates which way detected the error. Upper 1 bit unused. <p>Unaffected by Cold or Warm reset.</p> | |
| [27:26] | RESO | Reserved | 0b00 |
| [25] | SUBBANK | <p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L1 Instruction Cache]</p> <ul style="list-style-type: none"> Indicates which subbank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero. <p>Unaffected by Cold or Warm reset.</p> | |
| [24:23] | BANK | <p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L2 Cache]</p> <ul style="list-style-type: none"> Indicates which L2 bank detected the error. <p>[L1 Instruction Cache]</p> <ul style="list-style-type: none"> Indicates which bank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero. <p>Unaffected by Cold or Warm reset.</p> | |
| [22:19] | SUBARRAY | <p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L2 Cache]</p> <ul style="list-style-type: none"> Indicates which L2 data doubleword detected the error. Upper 1 bit is unused. <p>[L1 Data Cache]</p> <ul style="list-style-type: none"> Indicates for L1 Data RAM which word had the error detected. For L1 Tag RAMs which bank had the error (0b0000: bank0 , 0b0001: bank1) <p>Unaffected by Cold or Warm reset.</p> | |

| Bits | Name | Description | Reset |
|--------|-------|---|-------|
| [18:6] | INDEX | <p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L2 Cache]</p> <ul style="list-style-type: none"> Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. <p>[L1 Data Cache]</p> <ul style="list-style-type: none"> Indicates which index detected the error. Upper bits of the index are unused depending on the cache size <p>[L2 TLB]</p> <ul style="list-style-type: none"> Index of TLB RAM. Upper 4 bits are unused. <p>[L1 Instruction Cache]</p> <ul style="list-style-type: none"> Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. <p>Unaffected by Cold or Warm reset.</p> | |
| [5:4] | ARRAY | <p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L2 Cache]</p> <p>Indicates which array detected the error. The possible values are:</p> <ul style="list-style-type: none"> 0b00 L2 Tag RAM. 0b01 L2 Data RAM. 0b10 L2 TQ Data RAM. 0b11 CHI Error. <p>[L1 Data Cache]</p> <p>Indicates which array detected the error. The possible values are:</p> <ul style="list-style-type: none"> 00 LS Tag RAM 0. 01 LS Tag RAM 1. 10 LS Data RAM. 11 LS Tag RAM 2. <p>[L1 Instruction Cache]</p> <p>Indicates which array that detected the error, Data Array has higher priority. The possible values are:</p> <ul style="list-style-type: none"> 0b00 Tag. 0b01 Data. 0b10 Macro-OP cache. <p>Unaffected by Cold or Warm reset.</p> | |

| Bits | Name | Description | Reset |
|-------|------|--|-------|
| [3:0] | UNIT | Indicates the unit which detected the error. The possible values are: 0b0001 L1 Instruction Cache. 0b0010 L2 TLB. 0b0100 L1 Data Cache. 0b1000 L2 Cache. | |

Access

MRS <Xt>, ERXMISCO_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ERXMISCO_EL1 | 0b11 | 0b000 | 0b0101 | 0b0101 | 0b000 |

MSR ERXMISCO_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ERXMISCO_EL1 | 0b11 | 0b000 | 0b0101 | 0b0101 | 0b000 |

Accessibility

MRS <Xt>, ERXMISCO_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXMIScN_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISCO_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISCO_EL1;
elsif PSTATE.EL == EL3 then
    return ERXMISCO_EL1;

```

MSR ERXMISCO_EL1, <Xt>

```

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

```

```
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMIScN_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC0_EL1 = X[t];
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.TERR == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXMISC0_EL1 = X[t];
    elsif PSTATE.EL == EL3 then
        ERXMISC0_EL1 = X[t];
```

A.11.11 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1

Accesses ext-ERR<n>MISC1 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

0x0

Bit descriptions

Figure A-137: AArch64_erxmisc1_el1 bit assignments

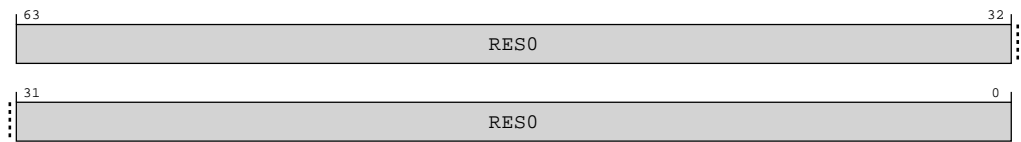


Table A-376: ERXMISC1_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, ERXMISC1_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ERXMISC1_EL1 | 0b11 | 0b000 | 0b0101 | 0b0101 | 0b001 |

MSR ERXMISC1_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ERXMISC1_EL1 | 0b11 | 0b000 | 0b0101 | 0b0101 | 0b001 |

Accessibility

MRS <Xt>, ERXMISC1_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXMISCn_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC1_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC1_EL1;
elseif PSTATE.EL == EL3 then
    return ERXMISC1_EL1;

```

MSR ERXMISC1_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC1_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC1_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXMISC1_EL1 = X[t];

```

A.11.12 ERXMISC2_EL1, Selected Error Record Miscellaneous Register 2

Accesses ext-ERR<n>MISC2 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

0x0

Bit descriptions

Figure A-138: AArch64_ermisc2_el1 bit assignments

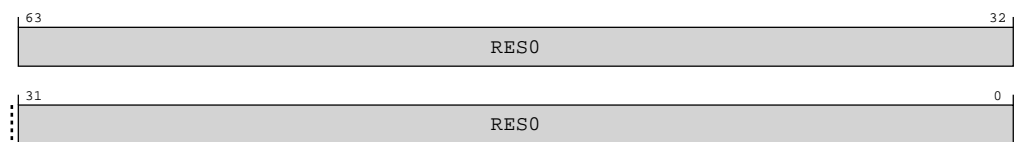


Table A-379: ERXMISC2_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, ERXMISC2_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ERXMISC2_EL1 | 0b11 | 0b000 | 0b0101 | 0b0101 | 0b010 |

MSR ERXMISC2_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ERXMISC2_EL1 | 0b11 | 0b000 | 0b0101 | 0b0101 | 0b010 |

Accessibility

MRS <Xt>, ERXMISC2_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```

```

    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCN_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC2_EL1;
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.TERR == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return ERXMISC2_EL1;
    elsif PSTATE.EL == EL3 then
        return ERXMISC2_EL1;

```

MSR ERXMISC2_EL1, <Xt>

```

    if PSTATE.EL == EL0 then
        UNDEFINED;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TERR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCN_EL1 == '1'
        then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif SCR_EL3.TERR == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXMISC2_EL1 = X[t];
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.TERR == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXMISC2_EL1 = X[t];
    elsif PSTATE.EL == EL3 then
        ERXMISC2_EL1 = X[t];

```

A.11.13 ERXMISC3_EL1, Selected Error Record Miscellaneous Register 3

Accesses ext-ERR<n>MISC3 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

0x0

Bit descriptions

Figure A-139: AArch64_erxmisc3_el1 bit assignments

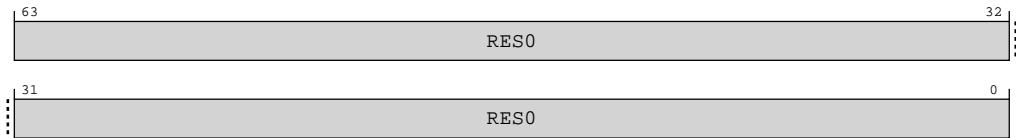


Table A-382: ERXMISC3_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [63:0] | RES0 | Reserved | 0x0 |

Access

MRS <Xt>, ERXMISC3_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ERXMISC3_EL1 | 0b11 | 0b000 | 0b0101 | 0b0101 | 0b011 |

MSR ERXMISC3_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|--------------|------|-------|--------|--------|-------|
| ERXMISC3_EL1 | 0b11 | 0b000 | 0b0101 | 0b0101 | 0b011 |

Accessibility

MRS <Xt>, ERXMISC3_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXMISCn_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC3_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC3_EL1;
elsif PSTATE.EL == EL3 then
    return ERXMISC3_EL1;

```

MSR ERXMISC3_EL1, <Xt>

```

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

```

```

if EL2Enabled() && HCR_EL2.TERR == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
then
    AArch64.SystemAccessTrap(EL2, 0x18);
elseif SCR_EL3.TERR == '1' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    ERXMISC3_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC3_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXMISC3_EL1 = X[t];

```

A.12 AArch64 Statistical Profiling Extension register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Statistical Profiling Extension registers in the core. For more information about a register, you can click the register name in the table.

Table A-385: AArch64 Statistical Profiling Extension register summary

| Name | Op0 | CRn | Op1 | CRm | Op2 | Reset | Width | Description |
|-----------------------------|-----|-----|-----|-----|-----|---------------------------|--------|--------------------------------|
| PMBIDR_EL1 | 3 | C9 | 0 | C10 | 7 | See individual bit resets | 64-bit | Profiling Buffer ID Register |
| PMSEVFR_EL1 | 3 | C9 | 0 | C9 | 5 | See individual bit resets | 64-bit | Sampling Event Filter Register |
| PMSIDR_EL1 | 3 | C9 | 0 | C9 | 7 | See individual bit resets | 64-bit | Sampling Profiling ID Register |

A.12.1 PMBIDR_EL1, Profiling Buffer ID Register

Provides information to software as to whether the buffer can be programmed at the current Exception level.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Statistical Profiling Extension

Reset value

See individual bit resets

Bit descriptions

Figure A-140: AArch64_pmbidr_el1 bit assignments

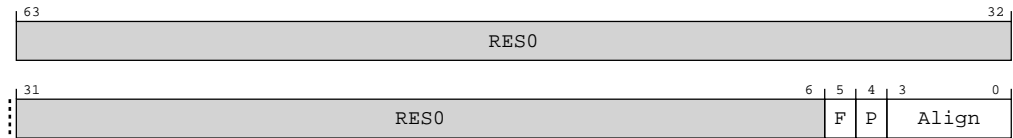


Table A-386: PMBIDR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|--|-------|
| [63:6] | RES0 | Reserved | 0x0 |
| [5] | F | Flag updates. Defines whether the address translation performed by the Profiling Buffer manages the Access Flag and dirty state. 0b1 Hardware management for the Access Flag and dirty state for accesses made by the Statistical Profiling Extension is controlled in the same way as explicit memory accesses in the owning translation regime. | |
| [4] | P | Programming not allowed. The Profiling Buffer is owned by a higher Exception level or the other Security state. 0b0 Profiling Buffer is owned by the current or a lower Exception level in the current Security state. | |
| [3:0] | Align | Defines the minimum alignment constraint for AArch64-PMBPTR_EL1. If this field is non-zero, then the PE must pad every record up to a multiple of this size. 0b0110 64 Bytes. | |

Access

MRS <Xt>, PMBIDR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| PMBIDR_EL1 | 0b11 | 0b000 | 0b1001 | 0b1010 | 0b111 |

Accessibility

MRS <Xt>, PMBIDR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    return PMBIDR_EL1;
elseif PSTATE.EL == EL2 then
    return PMBIDR_EL1;
elseif PSTATE.EL == EL3 then
    return PMBIDR_EL1;

```

A.12.2 PMSEVFR_EL1, Sampling Event Filter Register

Controls sample filtering by events. The overall filter is the logical AND of these filters. For example, if E[3] and E[5] are both set to 1, only samples that have both event 3 (Level 1 unified or data cache refill) and event 5 set (TLB walk) are recorded

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Statistical Profiling Extension

Reset value

See individual bit resets

Bit descriptions

Figure A-141: AArch64_pmsevfr_el1 bit assignments

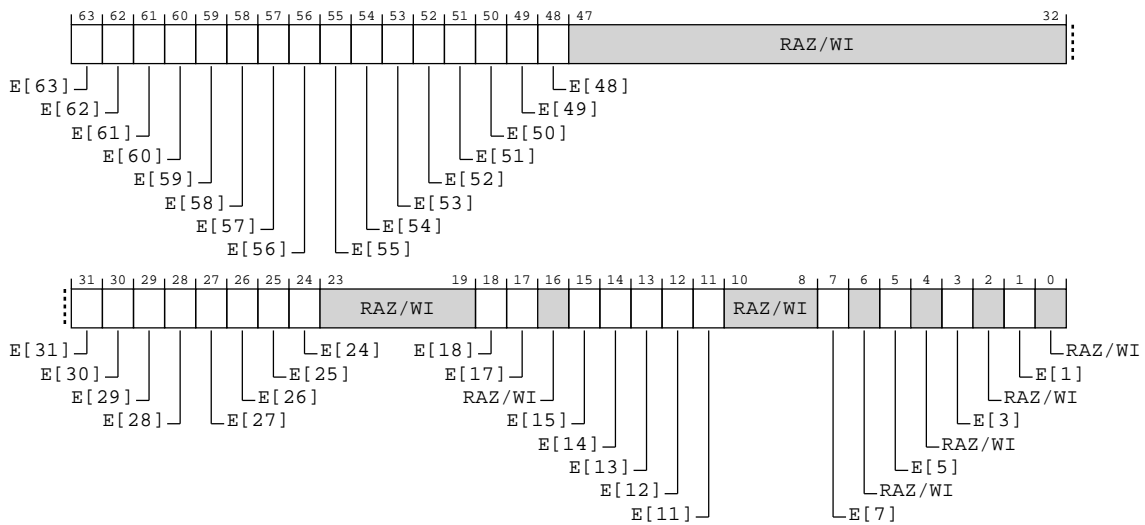


Table A-388: PMSEVFR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|------|-------|--|-------|
| [63] | E[63] | <p>E[63] is the event filter for event 63. If event 63 is not implemented, or filtering on event 63 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 63 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 63 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [62] | E[62] | <p>E[62] is the event filter for event 62. If event 62 is not implemented, or filtering on event 62 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 62 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 62 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [61] | E[61] | <p>E[61] is the event filter for event 61. If event 61 is not implemented, or filtering on event 61 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 61 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 61 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [60] | E[60] | <p>E[60] is the event filter for event 60. If event 60 is not implemented, or filtering on event 60 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 60 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 60 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |

| Bits | Name | Description | Reset |
|------|-------|--|-------|
| [59] | E[59] | <p>E[59] is the event filter for event 59. If event 59 is not implemented, or filtering on event 59 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 59 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 59 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [58] | E[58] | <p>E[58] is the event filter for event 58. If event 58 is not implemented, or filtering on event 58 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 58 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 58 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [57] | E[57] | <p>E[57] is the event filter for event 57. If event 57 is not implemented, or filtering on event 57 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 57 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 57 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [56] | E[56] | <p>E[56] is the event filter for event 56. If event 56 is not implemented, or filtering on event 56 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 56 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 56 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |

| Bits | Name | Description | Reset |
|------|-------|--|-------|
| [55] | E[55] | <p>E[55] is the event filter for event 55. If event 55 is not implemented, or filtering on event 55 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 55 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 55 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [54] | E[54] | <p>E[54] is the event filter for event 54. If event 54 is not implemented, or filtering on event 54 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 54 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 54 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [53] | E[53] | <p>E[53] is the event filter for event 53. If event 53 is not implemented, or filtering on event 53 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 53 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 53 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [52] | E[52] | <p>E[52] is the event filter for event 52. If event 52 is not implemented, or filtering on event 52 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 52 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 52 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |

| Bits | Name | Description | Reset |
|----------------------------------|------------|--|-------|
| [51] | E[51] | <p>E[51] is the event filter for event 51. If event 51 is not implemented, or filtering on event 51 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 51 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 51 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [50] | E[50] | <p>E[50] is the event filter for event 50. If event 50 is not implemented, or filtering on event 50 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 50 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 50 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [49] | E[49] | <p>E[49] is the event filter for event 49. If event 49 is not implemented, or filtering on event 49 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 49 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 49 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [48] | E[48] | <p>E[48] is the event filter for event 48. If event 48 is not implemented, or filtering on event 48 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 48 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 48 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [47:32, 23:19, 10:8, 6, 4, 2, 0] | RAZ/ WI | Reserved | |

| Bits | Name | Description | Reset |
|------|-------|--|-------|
| [31] | E[31] | <p>E[31] is the event filter for event 31. If event 31 is not implemented, or filtering on event 31 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 31 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 31 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [30] | E[30] | <p>E[30] is the event filter for event 30. If event 30 is not implemented, or filtering on event 30 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 30 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 30 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [29] | E[29] | <p>E[29] is the event filter for event 29. If event 29 is not implemented, or filtering on event 29 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 29 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 29 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [28] | E[28] | <p>E[28] is the event filter for event 28. If event 28 is not implemented, or filtering on event 28 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 28 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 28 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |

| Bits | Name | Description | Reset |
|------|-------|--|-------|
| [27] | E[27] | <p>E[27] is the event filter for event 27. If event 27 is not implemented, or filtering on event 27 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 27 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 27 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [26] | E[26] | <p>E[26] is the event filter for event 26. If event 26 is not implemented, or filtering on event 26 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 26 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 26 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [25] | E[25] | <p>E[25] is the event filter for event 25. If event 25 is not implemented, or filtering on event 25 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 25 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 25 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [24] | E[24] | <p>E[24] is the event filter for event 24. If event 24 is not implemented, or filtering on event 24 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 24 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 24 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |

| Bits | Name | Description | Reset |
|------|------------|--|-------|
| [18] | E[18] | <p>Empty predicate.</p> <p>0b0</p> <p>Empty predicate event is ignored.</p> <p>0b1</p> <p>Do not record samples that have the Empty predicate event == 0.</p> <p>This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.</p> | |
| [17] | E[17] | <p>Partial predicate.</p> <p>0b0</p> <p>Partial predicate event is ignored.</p> <p>0b1</p> <p>Do not record samples that have the Partial predicate event == 0.</p> <p>This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.</p> | |
| [16] | RAZ/ WI | Reserved | |
| [15] | E[15] | <p>E[15] is the event filter for event 15. If event 15 is not implemented, or filtering on event 15 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 15 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 15 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [14] | E[14] | <p>E[14] is the event filter for event 14. If event 14 is not implemented, or filtering on event 14 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 14 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 14 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |

| Bits | Name | Description | Reset |
|------|-------|--|-------|
| [13] | E[13] | <p>E[13] is the event filter for event 13. If event 13 is not implemented, or filtering on event 13 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 13 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 13 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [12] | E[12] | <p>E[12] is the event filter for event 12. If event 12 is not implemented, or filtering on event 12 is not supported, the corresponding bit is RAZ/WI.</p> <p>0b0</p> <p>Event 12 is ignored.</p> <p>0b1</p> <p>Do not record samples that have event 12 == 0.</p> <p>An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.</p> <p>This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0</p> | |
| [11] | E[11] | <p>Alignment.</p> <p>0b0</p> <p>Alignment event is ignored.</p> <p>0b1</p> <p>Do not record samples that have the Alignment event == 0.</p> <p>This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.</p> | |
| [7] | E[7] | <p>Mispredicted.</p> <p>0b0</p> <p>Mispredicted event is ignored.</p> <p>0b1</p> <p>Do not record samples that have the Mispredicted event == 0.</p> <p>This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.</p> | |
| [5] | E[5] | <p>TLB walk.</p> <p>0b0</p> <p>TLB walk event is ignored.</p> <p>0b1</p> <p>Do not record samples that have the TLB walk event == 0.</p> <p>This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.</p> | |

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [3] | E[3] | <p>Level 1 data or unified cache refill.</p> <p>0b0</p> <p>Level 1 data or unified cache refill event is ignored.</p> <p>0b1</p> <p>Do not record samples that have the Level 1 data or unified cache refill event == 0.</p> <p>This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.</p> | |
| [1] | E[1] | <p>Architecturally retired. When the PE supports sampling of speculative instructions:</p> <p>When the PE supports sampling of speculative instructions</p> <p>0b0</p> <p>Architecturally retired event is ignored.</p> <p>0b1</p> <p>Do not record samples that have the Architecturally retired event == 0.</p> | |

Access

MRS <Xt>, PMSEVFR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| PMSEVFR_EL1 | 0b11 | 0b000 | 0b1001 | 0b1001 | 0b101 |

MSR PMSEVFR_EL1, <Xt>

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| PMSEVFR_EL1 | 0b11 | 0b000 | 0b1001 | 0b1001 | 0b101 |

Accessibility

MRS <Xt>, PMSEVFR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMSEVFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        return NVMem[0x830];
    else
        return PMSEVFR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMSEVFR_EL1;
elsif PSTATE.EL == EL3 then
    return PMSEVFR_EL1;

```

MSR PMSEVFR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTen == '1' && HDFGWTR_EL2.PMSEVFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elseif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        NVMem[0x830] = X[t];
    else
        PMSEVFR_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elseif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSEVFR_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    PMSEVFR_EL1 = X[t];

```

A.12.3 PMSIDR_EL1, Sampling Profiling ID Register

Describes the Statistical Profiling implementation to software

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Statistical Profiling Extension

Reset value

See individual bit resets

Bit descriptions

Figure A-142: AArch64_pmsidr_el1 bit assignments

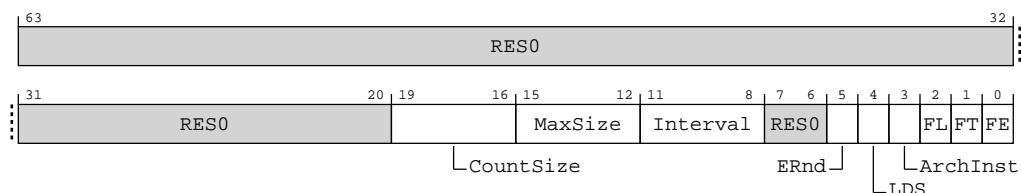


Table A-391: PMSIDR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|--------------|-----------|---|-------|
| [19:16] | CountSize | Defines the size of the counters 0b0010 12-bit saturating counters | |
| [15:12] | MaxSize | Defines the largest size for a single record, rounded up to a power-of-two. If this is the same as the minimum alignment (PMBIDR_EL1.Align), then each record is exactly this size 0b0110 64 bytes | |
| [11:8] | Interval | Recommended minimum sampling interval. This provides guidance from the implementer to the smallest minimum sampling interval, N. 0b0100 1,024 | |
| [5] | ERnd | Defines how the random number generator is used in determining the interval between samples, when enabled by PMSIRR_EL1.RND. 0b0 The random number is added at the start of the interval, and the sample is taken and a new interval started when the combined interval expires. | |
| [4] | LDS | Data source indicator for sampled load instructions 0b1 Loaded data source implemented | |
| [3] | ArchInst | Architectural instruction profiling 0b0 Micro-op sampling implemented | |
| [2] | FL | Filtering by latency. This bit reads as one. | |
| [1] | FT | Filtering by operation type. This bit reads as one. | |
| [0] | FE | Filtering by events. This bit reads as one. | |
| [63:20, 7:6] | RES0 | Reserved | 0b0 |

Access

MRS <Xt>, PMSIDR_EL1

| <systemreg> | op0 | op1 | CRn | CRm | op2 |
|-------------|------|-------|--------|--------|-------|
| PMSIDR_EL1 | 0b11 | 0b000 | 0b1001 | 0b1001 | 0b111 |

Accessibility

MRS <Xt>, PMSIDR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMSIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then

```

```
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMSIDR_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMSIDR_EL1;
elseif PSTATE.EL == EL3 then
    return PMSIDR_EL1;
```

Appendix B External registers

This appendix contains the descriptions for the Neoverse™ V2 core external or memory-mapped registers.

B.1 External CoreROM register summary

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

Table B-1: External CoreROM register summary

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

B.1.1 COREROM_ROMENTRY0, Core ROM table Entry 0

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

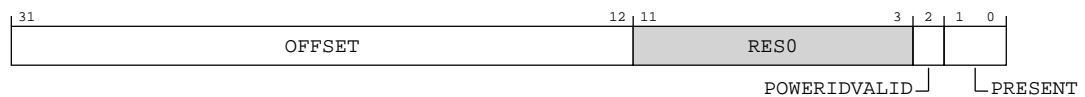
CoreROM

Register offset

0x000

Reset value

See individual bit resets

Bit descriptions**Figure B-1: ext_corerom_romentry0 bit assignments****Table B-2: COREROM_ROMENTRY0 bit descriptions**

| Bits | Name | Description | Reset |
|---------|--------------|--|-------|
| [31:12] | OFFSET | The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). 0b00000000000000000000000000000000 Core DBG component at address 0x1_0000. | |
| [11:3] | RES0 | Reserved | 0x0 |
| [2] | POWERIDVALID | Indicates if the Power domain ID field contains a Power domain ID. 0b0 A power domain ID is not provided. | |
| [1:0] | PRESENT | Indicates whether an entry is present at this location in the ROM Table. 0b11 The ROM Entry is present. | |

B.1.2 COREROM_ROMENTRY1, Core ROM table Entry 1

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

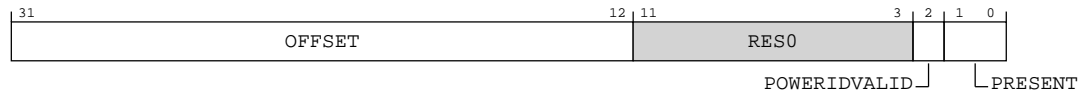
CoreROM

Register offset

0x004

Reset value

See individual bit resets

Bit descriptions**Figure B-2: ext_corerom_romentry1 bit assignments****Table B-3: COREROM_ROMENTRY1 bit descriptions**

| Bits | Name | Description | Reset |
|---------|--------------|---|-------|
| [31:12] | OFFSET | The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). 0b000000000000000100000 CORE PMU component at address 0x2_0000. | |
| [11:3] | RES0 | Reserved | 0x0 |
| [2] | POWERIDVALID | Indicates if the Power domain ID field contains a Power domain ID. 0b0 A power domain ID is not provided. | |
| [1:0] | PRESENT | Indicates whether an entry is present at this location in the ROM Table. 0b11 The ROM Entry is present. | |

B.1.3 COREROM_ROMENTRY2, Core ROM table Entry 2

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

CoreROM

Register offset

0x008

Reset value

See individual bit resets

Bit descriptions

Figure B-3: ext_corerom_romentry2 bit assignments

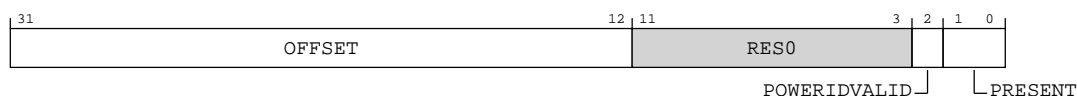


Table B-4: COREROM_ROMENTRY2 bit descriptions

| Bits | Name | Description | Reset |
|---------|--------------|--|-------|
| [31:12] | OFFSET | <p>The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation:</p> $\text{Component Address} = \text{ROM Table Base Address} + (\text{OFFSET} \ll 12).$ <p>0b000000000000000110000</p> <p>Core trace unit component at address 0x3_0000.</p> | |
| [11:3] | RES0 | Reserved | 0x0 |
| [2] | POWERIDVALID | <p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0</p> <p>A power domain ID is not provided.</p> | |
| [1:0] | PRESENT | <p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11</p> <p>The ROM Entry is present.</p> | |

B.1.4 COREROM_ROMENTRY3, Core ROM table Entry 3

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0x00C

Reset value

See individual bit resets

Bit descriptions

Figure B-4: ext_corerom_romentry3 bit assignments

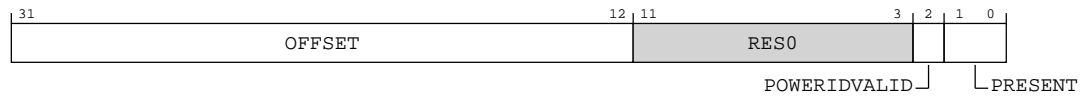


Table B-5: COREROM_ROMENTRY3 bit descriptions

| Bits | Name | Description | Reset |
|---------|--------------|---|-------|
| [31:12] | OFFSET | The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). 0b000000000000001000000 Core ELA component at address 0x4_0000. | |
| [11:3] | RES0 | Reserved | 0x0 |
| [2] | POWERIDVALID | Indicates if the Power domain ID field contains a Power domain ID. 0b0 A power domain ID is not provided. | |
| [1:0] | PRESENT | Indicates whether an entry is present at this location in the ROM Table. 0b11 The ROM Entry is present. | |

B.1.5 COREROM_AUTHSTATUS, Core ROM table Authentication Status Register

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

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFB8

Reset value

See individual bit resets

Bit descriptions

Figure B-5: ext_corerom_authstatus bit assignments

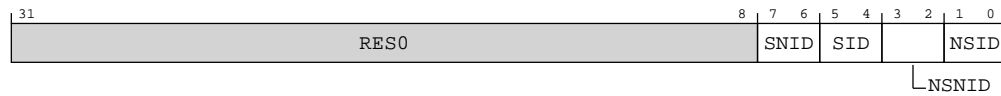


Table B-6: COREROM_AUTHSTATUS bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:6] | SNID | Secure Non-invasive Debug. ExternalSecureNoninvasiveDebugEnabled() == ExternalSecureInvasiveDebugEnabled(). This field has the same value as the SID field. | |
| [5:4] | SID | Secure Invasive Debug. 0b10 Secure invasive debug disabled. ExternalSecureInvasiveDebugEnabled() == FALSE. 0b11 Secure invasive debug enabled. ExternalSecureInvasiveDebugEnabled() == TRUE. | |
| [3:2] | NSNID | Non-secure Non-invasive Debug. 0b00 Debug level is not supported. | |
| [1:0] | NSID | Non-secure Invasive Debug. 0b00 Debug level is not supported. | |

B.1.6 COREROM_DEVARCH, Core ROM table Device Architecture Register

Identifies the architect and architecture of a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

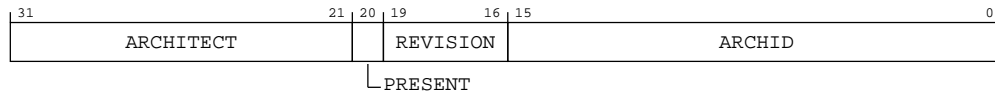
CoreROM

Register offset

0xFBC

Reset value

See individual bit resets

Bit descriptions**Figure B-6: ext_corerom_devarch bit assignments****Table B-7: COREROM_DEVARCH bit descriptions**

| Bits | Name | Description | Reset |
|---------|-----------|--|-------|
| [31:21] | ARCHITECT | Architect. 0b01000111011 JEP106 continuation code 0x4, ID code 0x3B. Arm Limited. | |
| [20] | PRESENT | Present. 0b1 DEVARCH information present. | |
| [19:16] | REVISION | Revision. 0b0000 Revision 0. | |
| [15:0] | ARCHID | Architecture ID. 0b0000101011110111 ROM Table v0. The debug tool must inspect ext-COREROM_DEVTYPE and ext-COREROM_DEVID to determine further information about the ROM Table. | |

B.1.7 COREROM_DEVTYPE, Core ROM table Device Type Register

A debugger can use DEVTYPE to obtain information about a component that has an unrecognized part number.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

CoreROM

Register offset

0xFCC

Reset value

See individual bit resets

Bit descriptions**Figure B-7: ext_corerom_devtype bit assignments****Table B-8: COREROM_DEVTYPE bit descriptions**

| Bits | Name | Description | Reset |
|--------|-------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | SUB | Sub number 0b0000 Other, undefined. | |
| [3:0] | MAJOR | Major number 0b0000 Miscellaneous. | |

B.1.8 COREROM_PIDR4, Core ROM table Peripheral Identification Register 4

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

CoreROM

Register offset

0xFD0

Reset value

See individual bit resets

Bit descriptions

Figure B-8: ext_corerom_pidr4 bit assignments

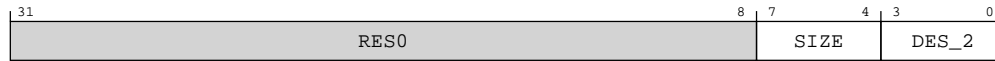


Table B-9: COREROM_PIDR4 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | SIZE | 4KB count. 0b0000 The component uses a single 4KB block. | |
| [3:0] | DES_2 | JEP106 continuation code. 0b0100 Arm Limited. Number of 0x7F bytes in full JEP106 code 0x7F 0x7F 0x7F 0x7F 0x3B. | |

B.1.9 COREROM_PIDR0, Core ROM table Peripheral Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFE0

Reset value

See individual bit resets

Bit descriptions

Figure B-9: ext_corerom_pidr0 bit assignments



Table B-10: COREROM_PIDR0 bit descriptions

| Bits | Name | Description | Reset |
|--------|--------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PART_0 | Part number bits [7:0]. 0b01001111 Neoverse™ V2 Core ROM table. Bits [7:0] of part number 0x4F. | |

B.1.10 COREROM_PIDR1, Core ROM table Peripheral Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFE4

Reset value

See individual bit resets

Bit descriptions

Figure B-10: ext_corerom_pidr1 bit assignments**Table B-11: COREROM_PIDR1 bit descriptions**

| Bits | Name | Description | Reset |
|--------|--------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | DES_0 | JEP106 identification code bits [3:0]. 0b1011 Arm Limited. Bits [3:0] of JEP106 identification code 0x3B. | |
| [3:0] | PART_1 | Part number bits [11:8]. 0b1101 Neoverse™ V2 Core ROM table. Bits [11:8] of part number 0x4F. | |

B.1.11 COREROM_PIDR2, Core ROM table Peripheral Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFE8

Reset value

See individual bit resets

Bit descriptions

Figure B-11: ext_corerom_pidr2 bit assignments



Table B-12: COREROM_PIDR2 bit descriptions

| Bits | Name | Description | Reset |
|--------|----------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | REVISION | Component revision. 0b0001 Revision rOp1. | |
| [3] | JEDEC | JEDEC assignee. 0b1 JEDEC-assignee values is used. | |
| [2:0] | DES_1 | JEP106 identification code bits [6:4]. 0b011 Arm Limited. Bits [6:4] of JEP106 identification code 0x3B. | |

B.1.12 COREROM_PIDR3, Core ROM table Peripheral Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFEC

Reset value

See individual bit resets

Bit descriptions

Figure B-12: ext_corerom_pidr3 bit assignments



Table B-13: COREROM_PIDR3 bit descriptions

| Bits | Name | Description | Reset |
|--------|--------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | REVAND | Minor errata fixes. 0b0000 No ECO fixes. | |
| [3:0] | CMOD | Customer Modified. 0b0000 The component is not modified from the original design. | |

B.1.13 COREROM_CIDR0, Core ROM table Component Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

CoreROM

Register offset

0xFF0

Reset value

See individual bit resets

Bit descriptions**Figure B-13: ext_corerom_cidr0 bit assignments****Table B-14: COREROM_CIDR0 bit descriptions**

| Bits | Name | Description | Reset |
|--------|---------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_0 | CoreSight component identification preamble. 0b00001101 CoreSight component identification preamble. | |

B.1.14 COREROM_CIDR1, Core ROM table Component Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

CoreROM

Register offset

0xFF4

Reset value

See individual bit resets

Bit descriptions

Figure B-14: ext_corerom_cidr1 bit assignments

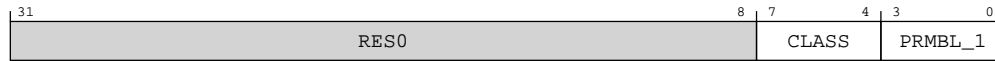


Table B-15: COREROM_CIDR1 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | CLASS | CoreSight component class. 0b1001 CoreSight component. | |
| [3:0] | PRMBL_1 | CoreSight component identification preamble. 0b0000 CoreSight component identification preamble. | |

B.1.15 COREROM_CIDR2, Core ROM table Component Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFF8

Reset value

See individual bit resets

Bit descriptions

Figure B-15: ext_corerom_cidr2 bit assignments



Table B-16: COREROM_CIDR2 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_2 | <p>CoreSight component identification preamble.</p> <p>0b00000101</p> <p>CoreSight component identification preamble.</p> | |

B.1.16 COREROM_CIDR3, Core ROM table Component Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFFC

Reset value

See individual bit resets

Bit descriptions

Figure B-16: ext_corerom_cidr3 bit assignments



Table B-17: COREROM_CIDR3 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_3 | CoreSight component identification preamble. 0b10110001 CoreSight component identification preamble. | |

B.2 External PPM register summary

The summary table provides an overview of all memory-mapped Power Performance Management (PPM) registers in the core. Individual register descriptions provide detailed information.

Table B-18: External PPM register summary

| Offset | Name | Reset | Width | Description |
|--------|---------------------------|---------------------------|--------|---------------------------------------|
| 0x000 | CPUPPMCR | See individual bit resets | 64-bit | Power Performance Management Register |
| 0x010 | CPUPPMCR2 | See individual bit resets | 64-bit | Power Performance Management Register |
| 0x020 | CPUPPMCR3 | See individual bit resets | 64-bit | Power Performance Management Register |
| 0x080 | CPUPPMCR4 | See individual bit resets | 64-bit | Power Performance Management Register |
| 0x088 | CPUPPMCR5 | See individual bit resets | 64-bit | Power Performance Management Register |
| 0x090 | CPUPPMCR6 | See individual bit resets | 64-bit | Power Performance Management Register |

B.2.1 CPUPPMCR, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x000

Reset value

See individual bit resets

Bit descriptions

Figure B-17: ext_cpupppmcr bit assignments

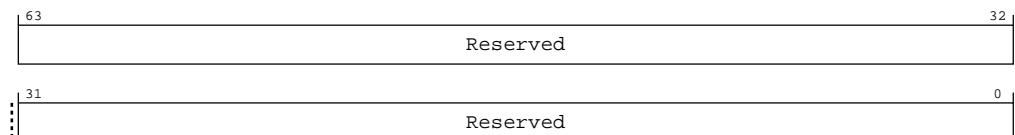


Table B-19: CPUPPMCR bit descriptions

| Bits | Name | Description | Reset |
|--------|----------|-------------|-------|
| [63:0] | Reserved | None | |

B.2.2 CPUPPMCR2, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

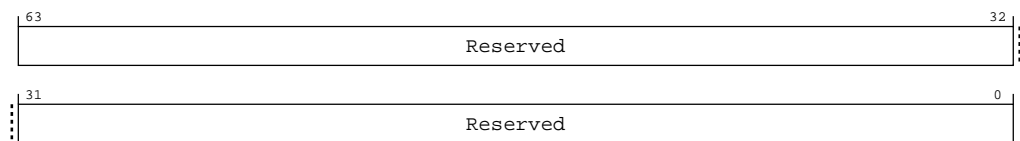
Register offset

0x010

Reset value

See individual bit resets

Bit descriptions

Figure B-18: ext_cpuppmcr2 bit assignments**Table B-20: CPUPPMCR2 bit descriptions**

| Bits | Name | Description | Reset |
|--------|----------|-------------|-------|
| [63:0] | Reserved | None | |

B.2.3 CPUPPMCR3, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes**Width**

64

Component

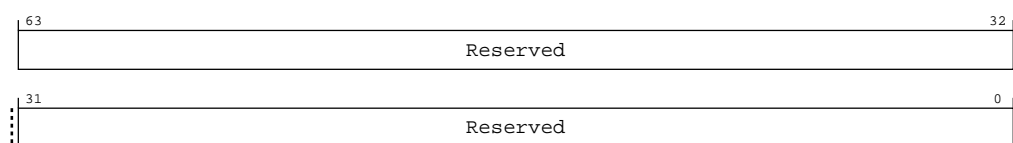
PPM

Register offset

0x020

Reset value

See individual bit resets

Bit descriptions**Figure B-19: ext_cpuppmcr3 bit assignments****Table B-21: CPUPPMCR3 bit descriptions**

| Bits | Name | Description | Reset |
|--------|----------|-------------|-------|
| [63:0] | Reserved | None | |

B.2.4 CPUPPMCR4, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes**Width**

64

Component

PPM

Register offset

0x080

Reset value

See individual bit resets

Bit descriptions

Figure B-20: ext_cpuppmcr4 bit assignments

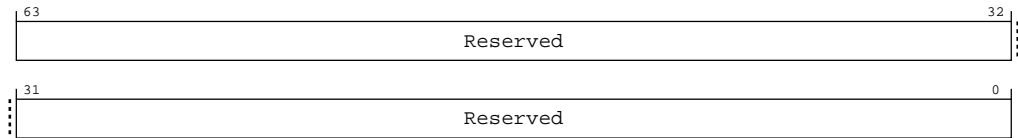


Table B-22: CPUPPMCR4 bit descriptions

| Bits | Name | Description | Reset |
|--------|----------|-------------|-------|
| [63:0] | Reserved | None | |

B.2.5 CPUPPMCR5, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x088

Reset value

See individual bit resets

Bit descriptions

Figure B-21: ext_cpuppmcr5 bit assignments

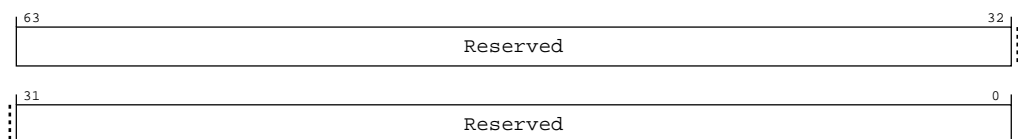


Table B-23: CPUPPMCR5 bit descriptions

| Bits | Name | Description | Reset |
|--------|----------|-------------|-------|
| [63:0] | Reserved | None | |

B.2.6 CPUPPMCR6, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x090

Reset value

See individual bit resets

Bit descriptions

Figure B-22: ext_cpuppmcr6 bit assignments

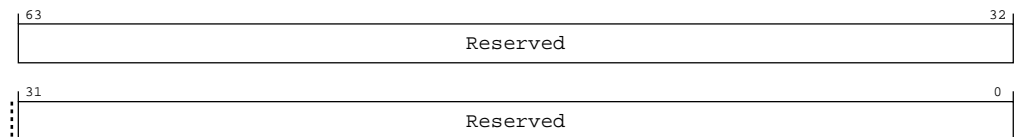


Table B-24: CPUPPMCR6 bit descriptions

| Bits | Name | Description | Reset |
|--------|----------|-------------|-------|
| [63:0] | Reserved | None | |

B.3 External PMU register summary

The summary table provides an overview of all memory-mapped performance monitors registers in the core. Individual register descriptions provide detailed information.

Table B-25: External PMU register summary

| Offset | Name | Reset | Width | Description |
|--------|---------------------------|---------------------------|--------|--|
| 0x600 | PMPCSSR | See individual bit resets | 64-bit | Snapshot Program Counter Sample Register |
| 0x608 | PMCIDSSR | See individual bit resets | 32-bit | Snapshot CONTEXTIDR_EL1 Sample Register |
| 0x60C | PMCID2SSR | See individual bit resets | 32-bit | Snapshot CONTEXTIDR_EL2 Sample Register |
| 0x610 | PMSSSR | See individual bit resets | 32-bit | PMU Snapshot Status Register |

| Offset | Name | Reset | Width | Description |
|--------|---------------------------|---------------------------|--------|---|
| 0x618 | PMCCNTSR | See individual bit resets | 64-bit | PMU Cycle Counter Snapshot Register |
| 0x620 | PMEVCNTR0 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x628 | PMEVCNTR1 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x630 | PMEVCNTR2 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x638 | PMEVCNTR3 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x640 | PMEVCNTR4 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x648 | PMEVCNTR5 | See individual bit resets | 64-bit | PMU Event Counter Snapshot Register |
| 0x6F0 | PMSSCR | See individual bit resets | 32-bit | PMU Snapshot Capture Register |
| 0xE00 | PMCFGR | See individual bit resets | 32-bit | Performance Monitors Configuration Register |
| 0xE04 | PMCR_ELO | See individual bit resets | 32-bit | Performance Monitors Control Register |
| 0xE20 | PMCEID0 | See individual bit resets | 32-bit | Performance Monitors Common Event Identification register 0 |
| 0xE24 | PMCEID1 | See individual bit resets | 32-bit | Performance Monitors Common Event Identification register 1 |
| 0xE28 | PMCEID2 | See individual bit resets | 32-bit | Performance Monitors Common Event Identification register 2 |
| 0xE2C | PMCEID3 | See individual bit resets | 32-bit | Performance Monitors Common Event Identification register 3 |
| 0xE40 | PMMIR | See individual bit resets | 32-bit | Performance Monitors Machine Identification Register |
| 0xFBC | PMDEVARCH | See individual bit resets | 32-bit | Performance Monitors Device Architecture register |
| 0xFC8 | PMDEVID | See individual bit resets | 32-bit | Performance Monitors Device ID register |
| 0xFCC | PMDEVTYPE | See individual bit resets | 32-bit | Performance Monitors Device Type register |
| 0xFD0 | PMPIDR4 | See individual bit resets | 32-bit | Performance Monitors Peripheral Identification Register 4 |
| 0xFE0 | PMPIDR0 | See individual bit resets | 32-bit | Performance Monitors Peripheral Identification Register 0 |
| 0xFE4 | PMPIDR1 | See individual bit resets | 32-bit | Performance Monitors Peripheral Identification Register 1 |
| 0xFE8 | PMPIDR2 | See individual bit resets | 32-bit | Performance Monitors Peripheral Identification Register 2 |
| 0xFEC | PMPIDR3 | See individual bit resets | 32-bit | Performance Monitors Peripheral Identification Register 3 |
| 0xFF0 | PMCIDR0 | See individual bit resets | 32-bit | Performance Monitors Component Identification Register 0 |
| 0xFF4 | PMCIDR1 | See individual bit resets | 32-bit | Performance Monitors Component Identification Register 1 |
| 0xFF8 | PMCIDR2 | See individual bit resets | 32-bit | Performance Monitors Component Identification Register 2 |
| 0xFFC | PMCIDR3 | See individual bit resets | 32-bit | Performance Monitors Component Identification Register 3 |

B.3.1 PMPCSSR, Snapshot Program Counter Sample Register

Captured copy of the Program Counter.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

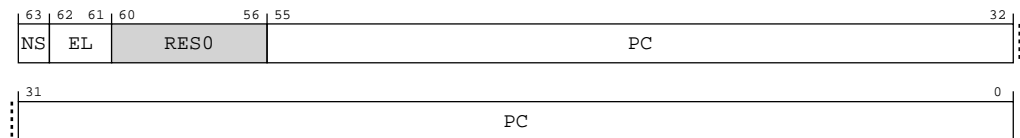
PMU

Register offset

0x600

Reset value

See individual bit resets

Bit descriptions**Figure B-23: ext_pmpcssr bit assignments****Table B-26: PMPCSSR bit descriptions**

| Bits | Name | Description | Reset |
|---------|------|--|---------|
| [63] | NS | Non-secure sample. 0b0 The captured instruction was executed in Secure state. 0b1 The captured instruction was executed in Non-secure state. | |
| [62:61] | EL | Exception level sample. The Exception level the captured instruction was executed at. | |
| [60:56] | RES0 | Reserved | 0b00000 |
| [55:0] | PC | Sampled PC. The instruction address for the sampled instruction. The sampled instruction must be an instruction recently executed by the PE. The architecture does not require that all instructions are eligible for sampling. However, it must be possible to reference instructions at branch targets. The branch target for a conditional branch instruction that fails its Condition code check is the instruction following the conditional branch target. The sampled instruction must be architecturally executed. However, in exceptional circumstances, such as a change in security state or other boundary condition, it is permissible to sample an instruction that was speculatively executed and not architecturally executed. When <i>Embedded Trace Extension</i> (ETE) is enabled, this register only captures branch targets. This register captures the target of the last branch retired similar to previous generations. When ETE is disabled, this register captures the PC of the last instruction retired. This functionality allows sampling of all instructions. Note: The Arm architecture does not define recently executed. | |

B.3.2 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register

Captured copy of the CONTEXTIDR_EL1 register.

The value captured must relate to the instruction captured in PMPCSSR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x608

Reset value

See individual bit resets

Bit descriptions

Figure B-24: ext_pmcidssr bit assignments

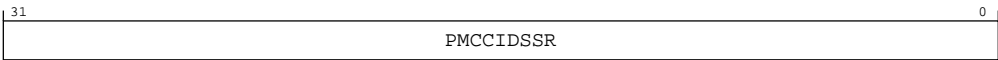


Table B-27: PMCIDSSR bit descriptions

| Bits | Name | Description | Reset |
|--------|-----------|--|-------|
| [31:0] | PMCCIDSSR | PMCIDSR sample. Sampled CONTEXTIDR_EL1 snapshot. | |

B.3.3 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register

Captured copy of the CONTEXTIDR_EL2 register.

The value captured must relate to the instruction captured in PMPCSSR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

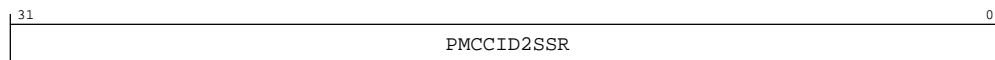
PMU

Register offset

0x60C

Reset value

See individual bit resets

Bit descriptions**Figure B-25: ext_pmcid2ssr bit assignments****Table B-28: PMCID2SSR bit descriptions**

| Bits | Name | Description | Reset |
|--------|------------|---|-------|
| [31:0] | PMCCID2SSR | PMCID2SR sample. Sampled CONTEXTIDR_EL2 snapshot. | |

B.3.4 PMSSSR, PMU Snapshot Status Register

Holds status information about the captured counters.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

PMU

Register offset

0x610

Reset value

See individual bit resets

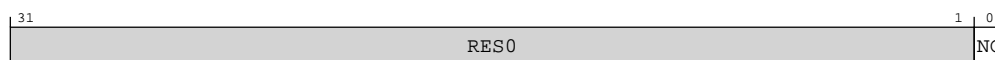
Bit descriptions**Figure B-26: ext_pmsssr bit assignments**

Table B-29: PMSSSR bit descriptions

| Bits | Name | Description | Reset |
|--------|------|--|-------|
| [31:1] | RES0 | Reserved | 0x0 |
| [0] | NC | <p>No capture. Indicates whether the PMU counters have been captured.</p> <p>0b0</p> <p>PMU counters captured.</p> <p>0b1</p> <p>PMU counters not captured.</p> <p>The event counters are only not captured by the PE in the event of a security violation. The external Monitor is responsible for keeping track of whether it managed to capture the snapshot registers from the PE.</p> <p>PMSSR.NC does not reflect the status of the captured Program Counter Sample registers.</p> <p>PMSSR.NC is reset to 1 by PE Warm reset, but is overwritten at the first capture. Tools need to be aware that capturing over reset or power-down might lose data, as they are reliant on software saving and restoring the PMU state (including PMSSCR). There is no sampled sticky reset bit.</p> | |

B.3.5 PMCCNTSR, PMU Cycle Counter Snapshot Register

Captured copy of PMCCNTR_ELO. Once captured, the value in PMCCNTSR is unaffected by writes to PMCCNTR_ELO and PMCR_ELO.C.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x618

Reset value

See individual bit resets

Bit descriptions

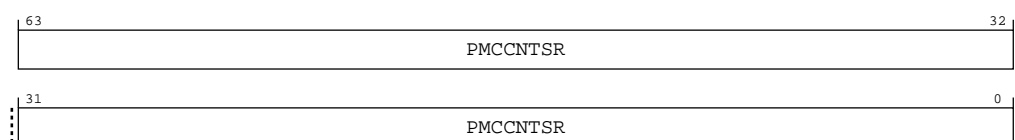
Figure B-27: ext_pmccntsr bit assignments

Table B-30: PMCCNTR bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|--|-------|
| [63:0] | PMCCNTR | PMCCNTR_ELO sample. Sampled cycle count. | |

B.3.6 PMEVCNTR0, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_ELO and PMCR_ELO.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

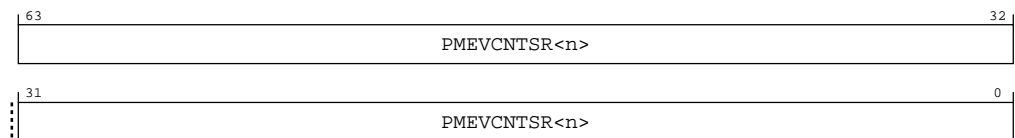
Register offset

0x620

Reset value

See individual bit resets

Bit descriptions

Figure B-28: ext_pmevcntr0 bit assignments**Table B-31: PMEVCNTR0 bit descriptions**

| Bits | Name | Description | Reset |
|--------|-------------|--|-------|
| [63:0] | PMEVCNTR<n> | PMEVCNTR<n>_ELO sample. Sampled event count. | |

B.3.7 PMEVCNTR1, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_ELO and PMCR_ELO.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x628

Reset value

See individual bit resets

Bit descriptions

Figure B-29: ext_pmevcntrs1 bit assignments

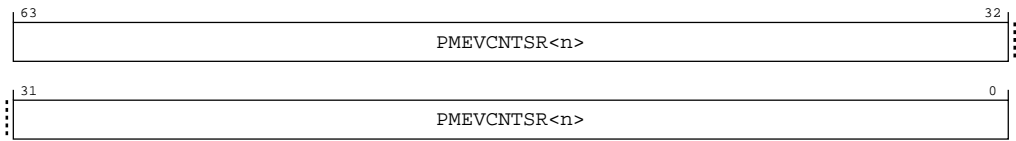


Table B-32: PMEVCNTR1 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------------|--|-------|
| [63:0] | PMEVCNTR<n> | PMEVCNTR<n>_ELO sample. Sampled event count. | |

B.3.8 PMEVCNTR2, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_ELO and PMCR_ELO.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x630

Reset value

See individual bit resets

Bit descriptions

Figure B-30: ext_pmevcntr2 bit assignments

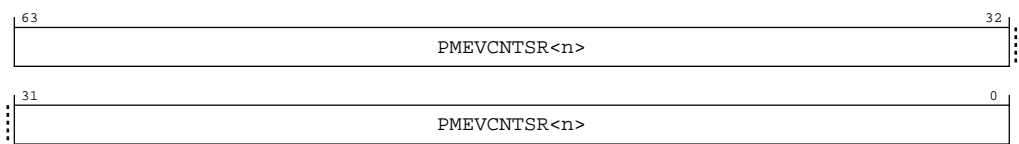


Table B-33: PMEVCNTR2 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------------|--|-------|
| [63:0] | PMEVCNTR<n> | PMEVCNTR<n>_ELO sample. Sampled event count. | |

B.3.9 PMEVCNTR3, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_ELO and PMCR_ELO.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x638

Reset value

See individual bit resets

Bit descriptions

Figure B-31: ext_pmevcntr3 bit assignments

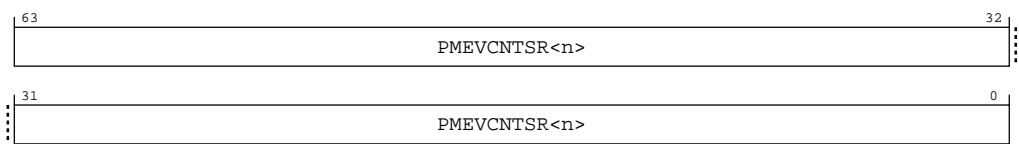


Table B-34: PMEVCNTR3 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------------|--|-------|
| [63:0] | PMEVCNTR<n> | PMEVCNTR<n>_ELO sample. Sampled event count. | |

B.3.10 PMEVCNTR4, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_ELO and PMCR_ELO.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

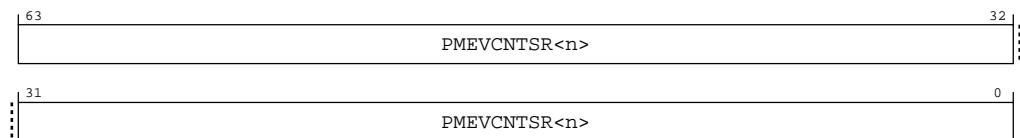
Register offset

0x640

Reset value

See individual bit resets

Bit descriptions

Figure B-32: ext_pmevcntr4 bit assignments**Table B-35: PMEVCNTR4 bit descriptions**

| Bits | Name | Description | Reset |
|--------|-------------|--|-------|
| [63:0] | PMEVCNTR<n> | PMEVCNTR<n>_ELO sample. Sampled event count. | |

B.3.11 PMEVCNTR5, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_ELO and PMCR_ELO.P.

Configurations

This register is available in all configurations.

Attributes**Width**

64

Component

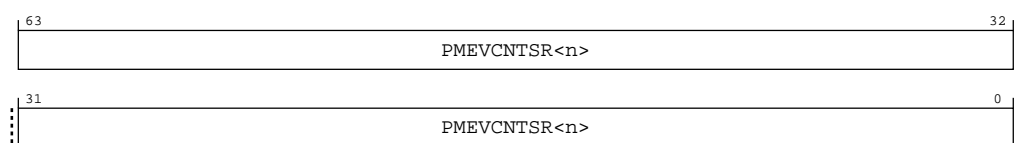
PMU

Register offset

0x648

Reset value

See individual bit resets

Bit descriptions**Figure B-33: ext_pmevcntrs5 bit assignments****Table B-36: PMEVCNTR5 bit descriptions**

| Bits | Name | Description | Reset |
|--------|-------------|--|-------|
| [63:0] | PMEVCNTR<n> | PMEVCNTR<n>_ELO sample. Sampled event count. | |

B.3.12 PMSSCR, PMU Snapshot Capture Register

Provides a mechanism for software to initiate a sample.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

PMU

Register offset

0x6F0

Reset value

See individual bit resets

Bit descriptions

Figure B-34: ext_pmsscr bit assignments

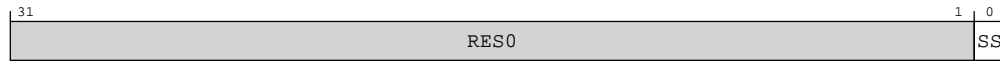


Table B-37: PMSSCR bit descriptions

| Bits | Name | Description | Reset |
|--------|------|--|-------|
| [31:1] | RES0 | Reserved | 0x0 |
| [0] | SS | <p>Capture now.</p> <p>0b0 Ignored.</p> <p>0b1 Initiate a capture immediately.</p> | |

B.3.13 PMCFGR, Performance Monitors Configuration Register

Contains PMU-specific configuration data.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE00

Reset value

See individual bit resets

Bit descriptions

Figure B-35: ext_pmcfggr bit assignments

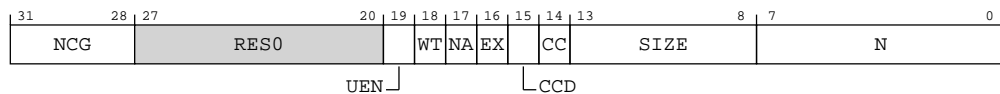


Table B-38: PMCFGR bit descriptions

| Bits | Name | Description | Reset |
|---------|------|---|------------|
| [31:28] | NCG | This feature is not supported, so this field is RAZ. | |
| [27:20] | RES0 | Reserved | 0b00000000 |
| [19] | UEN | User-mode Enable Register supported. AArch64-PMUSERENR_ELO is not visible in the external debug interface, so this bit is RAZ. | |
| [18] | WT | This feature is not supported, so this bit is RAZ. | |
| [17] | NA | This feature is not supported, so this bit is RAZ. | |
| [16] | EX | Export not supported. 0b0 ext-PMCR_ELO.X is RES0. | |
| [15] | CCD | Cycle counter has prescale. This bit is RAZ because this core does not support AArch32. | |
| [14] | CC | Dedicated cycle counter (counter 31) supported. This bit is RAO. | |
| [13:8] | SIZE | Size of counters, minus one. This field defines the size of the largest counter implemented by the Performance Monitors Unit. From Armv8, the largest counter is 64-bits, so the value of this field is 0b111111. This field is used by software to determine the spacing of the counters in the memory-map. From Armv8, the counters are a doubleword-aligned addresses. | |
| [7:0] | N | Number of counters implemented in addition to the cycle counter, ext-PMCCNTR_ELO. 0b00000110 ext-PMCCNTR_ELO plus six event counters implemented. | |

B.3.14 PMCR_ELO, Performance Monitors Control Register

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE04

Reset value

See individual bit resets

B.3.15 PMCEID0, Performance Monitors Common Event Identification register 0

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x0000 to 0x001F

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'. - Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. - This view of the register was previously called PMCEID0_ELO.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE20

Reset value

See individual bit resets

Bit descriptions

Figure B-37: ext_pmceid0 bit assignments

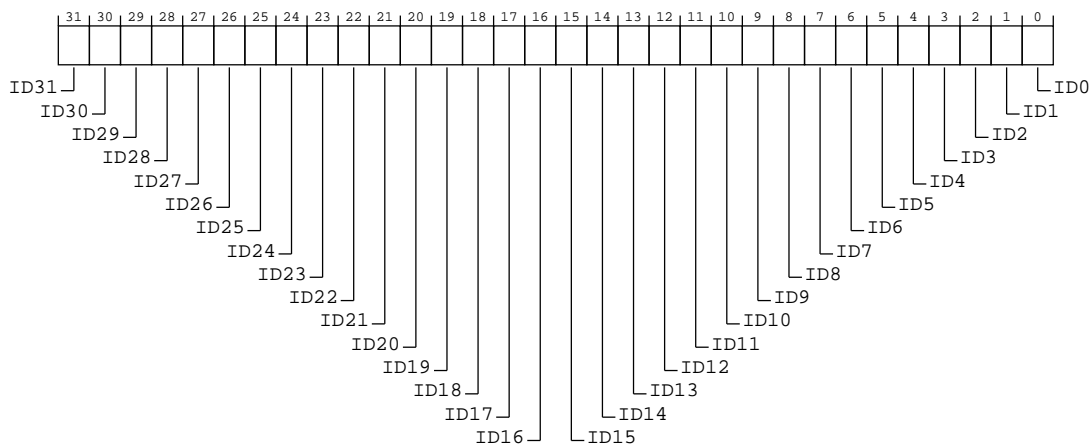


Table B-40: PMCEID0 bit descriptions

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [31] | ID31 | ID31 corresponds to common event (0x1f) L1D_CACHE_ALLOCATE 0b0 The common event is not implemented, or not counted. | |
| [30] | ID30 | ID30 corresponds to common event (0x1e) CHAIN 0b1 The common event is implemented. | |
| [29] | ID29 | ID29 corresponds to common event (0x1d) BUS_CYCLES 0b1 The common event is implemented. | |
| [28] | ID28 | ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED 0b1 The common event is implemented. | |
| [27] | ID27 | ID27 corresponds to common event (0x1b) INST_SPEC 0b1 The common event is implemented. | |
| [26] | ID26 | ID26 corresponds to common event (0x1a) MEMORY_ERROR 0b1 The common event is implemented. | |
| [25] | ID25 | ID25 corresponds to common event (0x19) BUS_ACCESS 0b1 The common event is implemented. | |
| [24] | ID24 | ID24 corresponds to common event (0x18) L2D_CACHE_WB 0b1 The common event is implemented. | |
| [23] | ID23 | ID23 corresponds to common event (0x17) L2D_CACHE_REFILL 0b1 The common event is implemented. | |
| [22] | ID22 | ID22 corresponds to common event (0x16) L2D_CACHE 0b1 The common event is implemented. | |
| [21] | ID21 | ID21 corresponds to common event (0x15) L1D_CACHE_WB 0b1 The common event is implemented. | |
| [20] | ID20 | ID20 corresponds to common event (0x14) L1I_CACHE 0b1 The common event is implemented. | |
| [19] | ID19 | ID19 corresponds to common event (0x13) MEM_ACCESS 0b1 The common event is implemented. | |

| Bits | Name | Description | Reset |
|------|------|---|-------|
| [18] | ID18 | ID18 corresponds to common event (0x12) BR_PRED 0b1 The common event is implemented. | |
| [17] | ID17 | ID17 corresponds to common event (0x11) CPU_CYCLES 0b1 The common event is implemented. | |
| [16] | ID16 | ID16 corresponds to common event (0x10) BR_MIS_PRED 0b1 The common event is implemented. | |
| [15] | ID15 | ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [14] | ID14 | ID14 corresponds to common event (0xe) BR_RETURN_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [13] | ID13 | ID13 corresponds to common event (0xd) BR_IMMED_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [12] | ID12 | ID12 corresponds to common event (0xc) PC_WRITE_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [11] | ID11 | ID11 corresponds to common event (0xb) CID_WRITE_RETIRED 0b1 The common event is implemented. | |
| [10] | ID10 | ID10 corresponds to common event (0xa) EXC_RETURN 0b1 The common event is implemented. | |
| [9] | ID9 | ID9 corresponds to common event (0x9) EXC_TAKEN 0b1 The common event is implemented. | |
| [8] | ID8 | ID8 corresponds to common event (0x8) INST_RETIRED 0b1 The common event is implemented. | |
| [7] | ID7 | ID7 corresponds to common event (0x7) ST_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [6] | ID6 | ID6 corresponds to common event (0x6) LD_RETIRED 0b0 The common event is not implemented, or not counted. | |
| [5] | ID5 | ID5 corresponds to common event (0x5) L1D_TLB_REFILL 0b1 The common event is implemented. | |

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [4] | ID4 | ID4 corresponds to common event (0x4) L1D_CACHE 0b1 The common event is implemented. | |
| [3] | ID3 | ID3 corresponds to common event (0x3) L1D_CACHE_REFILL 0b1 The common event is implemented. | |
| [2] | ID2 | ID2 corresponds to common event (0x2) L1I_TLB_REFILL 0b1 The common event is implemented. | |
| [1] | ID1 | ID1 corresponds to common event (0x1) L1I_CACHE_REFILL 0b1 The common event is implemented. | |
| [0] | ID0 | ID0 corresponds to common event (0x0) SW_INCR 0b1 The common event is implemented. | |

B.3.16 PMCEID1, Performance Monitors Common Event Identification register 1

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

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'. - Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. - This view of the register was previously called PMCEID1_ELO.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE24

Reset value

See individual bit resets

Bit descriptions

Figure B-38: ext_pmceid1 bit assignments

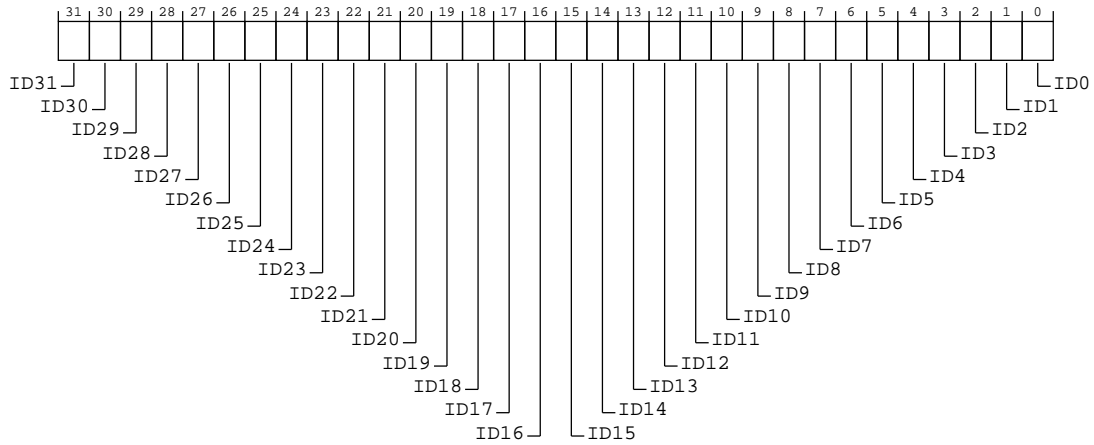


Table B-41: PMCEID1 bit descriptions

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [31] | ID31 | ID31 corresponds to common event (0x3f) STALL_SLOT 0b1 The common event is implemented. | |
| [30] | ID30 | ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND 0b1 The common event is implemented. | |
| [29] | ID29 | ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND 0b1 The common event is implemented. | |
| [28] | ID28 | ID28 corresponds to common event (0x3c) STALL 0b1 The common event is implemented. | |
| [27] | ID27 | ID27 corresponds to common event (0x3b) OP_SPEC 0b1 The common event is implemented. | |
| [26] | ID26 | ID26 corresponds to common event (0x3a) OP_RETIRED 0b1 The common event is implemented. | |
| [25] | ID25 | ID25 corresponds to common event (0x39) L1D_CACHE_LMISS_RD 0b1 The common event is implemented. | |
| [24] | ID24 | ID24 corresponds to common event (0x38) REMOTE_ACCESS_RD 0b0 The common event is not implemented, or not counted. | |

| Bits | Name | Description | Reset |
|------|------|--|-------|
| [23] | ID23 | ID23 corresponds to common event (0x37) LL_CACHE_MISS_RD 0b1 The common event is implemented. | |
| [22] | ID22 | ID22 corresponds to common event (0x36) LL_CACHE_RD 0b1 The common event is implemented. | |
| [21] | ID21 | ID21 corresponds to common event (0x35) ITLB_WLK 0b1 The common event is implemented. | |
| [20] | ID20 | ID20 corresponds to common event (0x34) DTLB_WLK 0b1 The common event is implemented. | |
| [19] | ID19 | ID19 corresponds to a Reserved Event event (0x33) 0b0 The common event is not implemented, or not counted. | |
| [18] | ID18 | ID18 corresponds to a Reserved Event event (0x32) 0b0 The common event is not implemented, or not counted. | |
| [17] | ID17 | ID17 corresponds to common event (0x31) REMOTE_ACCESS 0b1 The common event is implemented. | |
| [16] | ID16 | ID16 corresponds to common event (0x30) L2I_TLB 0b0 The common event is not implemented, or not counted. | |
| [15] | ID15 | ID15 corresponds to common event (0x2f) L2TLB_REQ 0b1 The common event is implemented. | |
| [14] | ID14 | ID14 corresponds to common event (0x2e) L2I_TLB_REFILL 0b0 The common event is not implemented, or not counted. | |
| [13] | ID13 | ID13 corresponds to common event (0x2d) L2TLB_REFILL 0b1 The common event is implemented. | |
| [12] | ID12 | ID12 corresponds to common event (0x2c) Reserved 0b0 The common event is not implemented, or not counted. | |
| [11] | ID11 | ID11 corresponds to common event (0x2b) L3D_CACHE 0b1 The common event is implemented. | |
| [10] | ID10 | ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL 0b1 The common event is implemented. | |

| Bits | Name | Description | Reset |
|------|------|---|-------|
| [9] | ID9 | ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE 0b1 The common event is implemented. | |
| [8] | ID8 | ID8 corresponds to common event (0x28) L2I_CACHE_REFILL 0b0 The common event is not implemented, or not counted. | |
| [7] | ID7 | ID7 corresponds to common event (0x27) L2I_CACHE 0b0 The common event is not implemented, or not counted. | |
| [6] | ID6 | ID6 corresponds to common event (0x26) L1I_TLB 0b1 The common event is implemented. | |
| [5] | ID5 | ID5 corresponds to common event (0x25) L1D_TLB 0b1 The common event is implemented. | |
| [4] | ID4 | ID4 corresponds to common event (0x24) STALL_BACKEND 0b1 The common event is implemented. | |
| [3] | ID3 | ID3 corresponds to common event (0x23) STALL_FRONTEND 0b1 The common event is implemented. | |
| [2] | ID2 | ID2 corresponds to common event (0x22) BR_MIS_PRED_RETIRE 0b1 The common event is implemented. | |
| [1] | ID1 | ID1 corresponds to common event (0x21) BR_RETIRE 0b1 The common event is implemented. | |
| [0] | ID0 | ID0 corresponds to common event (0x20) L2D_CACHE_ALLOCATE 0b1 The common event is implemented. | |

B.3.17 PMCEID2, Performance Monitors Common Event Identification register 2

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

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE28

Reset value

See individual bit resets

Bit descriptions

Figure B-39: ext_pmceid2 bit assignments

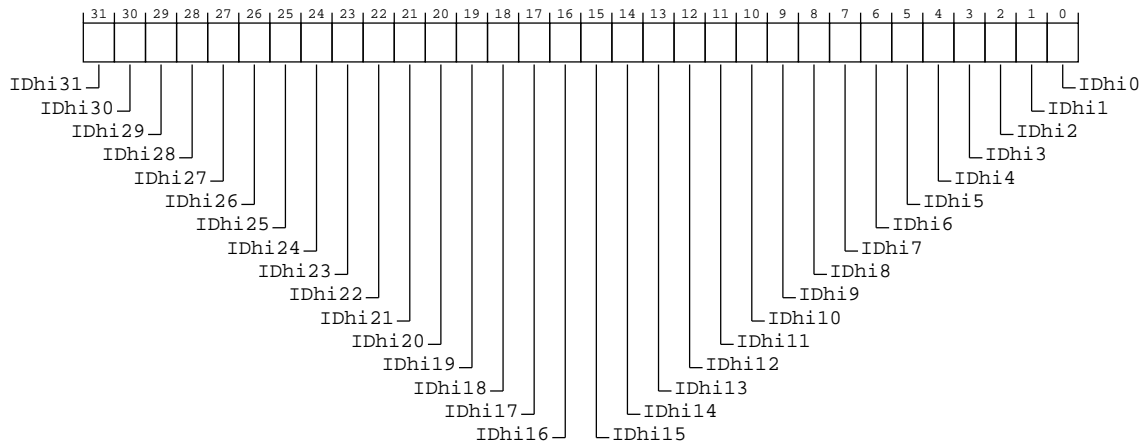


Table B-42: PMCEID2 bit descriptions

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [31] | IDHi31 | IDHi31 corresponds to a Reserved Event event (0x401f) 0b0 The common event is not implemented, or not counted. | |
| [30] | IDHi30 | IDHi30 corresponds to a Reserved Event event (0x401e) 0b0 The common event is not implemented, or not counted. | |
| [29] | IDHi29 | IDHi29 corresponds to a Reserved Event event (0x401d) 0b0 The common event is not implemented, or not counted. | |

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [28] | IDhi28 | IDhi28 corresponds to a Reserved Event event (0x401c) 0b0 The common event is not implemented, or not counted. | |
| [27] | IDhi27 | IDhi27 corresponds to common event (0x401b) CTI_TRIGOUT7 0b1 The common event is implemented. | |
| [26] | IDhi26 | IDhi26 corresponds to common event (0x401a) CTI_TRIGOUT6 0b1 The common event is implemented. | |
| [25] | IDhi25 | IDhi25 corresponds to common event (0x4019) CTI_TRIGOUT5 0b1 The common event is implemented. | |
| [24] | IDhi24 | IDhi24 corresponds to common event (0x4018) CTI_TRIGOUT4 0b1 The common event is implemented. | |
| [23] | IDhi23 | IDhi23 corresponds to a Reserved Event event (0x4017) 0b0 The common event is not implemented, or not counted. | |
| [22] | IDhi22 | IDhi22 corresponds to a Reserved Event event (0x4016) 0b0 The common event is not implemented, or not counted. | |
| [21] | IDhi21 | IDhi21 corresponds to a Reserved Event event (0x4015) 0b0 The common event is not implemented, or not counted. | |
| [20] | IDhi20 | IDhi20 corresponds to a Reserved Event event (0x4014) 0b0 The common event is not implemented, or not counted. | |
| [19] | IDhi19 | IDhi19 corresponds to common event (0x4013) TRCEXTOUT3 0b1 The common event is implemented. | |
| [18] | IDhi18 | IDhi18 corresponds to common event (0x4012) TRCEXTOUT2 0b1 The common event is implemented. | |
| [17] | IDhi17 | IDhi17 corresponds to common event (0x4011) TRCEXTOUT1 0b1 The common event is implemented. | |
| [16] | IDhi16 | IDhi16 corresponds to common event (0x4010) TRCEXTOUT0 0b1 The common event is implemented. | |
| [15] | IDhi15 | IDhi15 corresponds to common event (0x400f) Reserved 0b0 The common event is not implemented, or not counted. | |

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [14] | IDhi14 | IDhi14 corresponds to common event (0x400e) TRB_TRIG 0b0 The common event is not implemented, or not counted. | |
| [13] | IDhi13 | IDhi13 corresponds to common event (0x400d) PMU_OVFS 0b0 The common event is not implemented, or not counted. | |
| [12] | IDhi12 | IDhi12 corresponds to common event (0x400c) TRB_WRAP 0b1 The common event is implemented. | |
| [11] | IDhi11 | IDhi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD 0b1 The common event is implemented. | |
| [10] | IDhi10 | IDhi10 corresponds to common event (0x400a) L2I_CACHE_LMISS 0b0 The common event is not implemented, or not counted. | |
| [9] | IDhi9 | IDhi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD 0b1 The common event is implemented. | |
| [8] | IDhi8 | IDhi8 corresponds to common event (0x4008) Reserved 0b0 The common event is not implemented, or not counted. | |
| [7] | IDhi7 | IDhi7 corresponds to common event (0x4007) Reserved 0b0 The common event is not implemented, or not counted. | |
| [6] | IDhi6 | IDhi6 corresponds to common event (0x4006) L1I_CACHE_LMISS 0b1 The common event is implemented. | |
| [5] | IDhi5 | IDhi5 corresponds to common event (0x4005) STALL_BACKEND_MEM 0b1 The common event is implemented. | |
| [4] | IDhi4 | IDhi4 corresponds to common event (0x4004) CNT_CYCLES 0b1 The common event is implemented. | |
| [3] | IDhi3 | IDhi3 corresponds to common event (0x4003) SAMPLE_COLLISION 0b1 The common event is implemented. | |
| [2] | IDhi2 | IDhi2 corresponds to common event (0x4002) SAMPLE_FILTRATE 0b1 The common event is implemented. | |
| [1] | IDhi1 | IDhi1 corresponds to common event (0x4001) SAMPLE_FEED 0b1 The common event is implemented. | |

| Bits | Name | Description | Reset |
|------|-------|---|-------|
| [0] | IDhi0 | IDhi0 corresponds to common event (0x4000) SAMPLE_POP 0b1 The common event is implemented. | |

B.3.18 PMCEID3, Performance Monitors Common Event Identification register 3

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

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE2C

Reset value

See individual bit resets

Bit descriptions

Figure B-40: ext_pmceid3 bit assignments

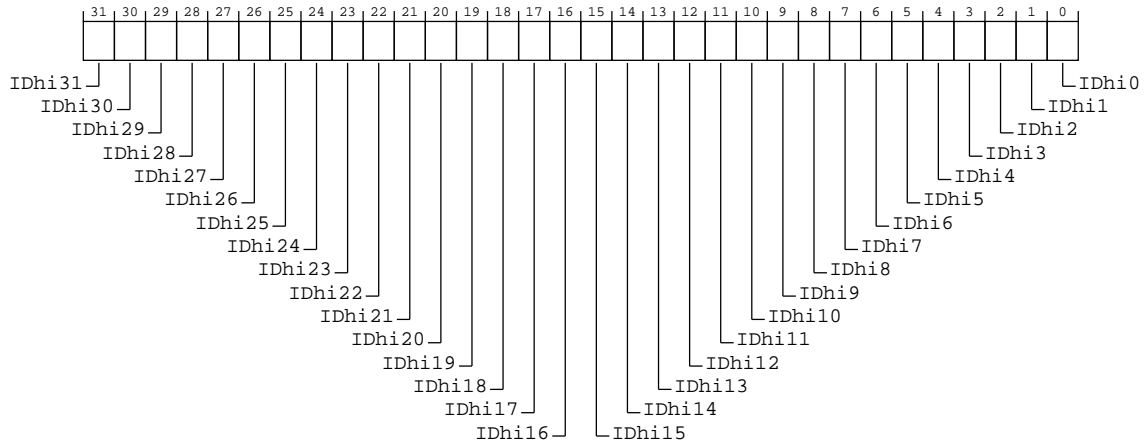


Table B-43: PMCEID3 bit descriptions

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [31] | IDhi31 | IDhi31 corresponds to a Reserved Event event (0x403f) 0b0 The common event is not implemented, or not counted. | |
| [30] | IDhi30 | IDhi30 corresponds to a Reserved Event event (0x403e) 0b0 The common event is not implemented, or not counted. | |
| [29] | IDhi29 | IDhi29 corresponds to a Reserved Event event (0x403d) 0b0 The common event is not implemented, or not counted. | |
| [28] | IDhi28 | IDhi28 corresponds to a Reserved Event event (0x403c) 0b0 The common event is not implemented, or not counted. | |
| [27] | IDhi27 | IDhi27 corresponds to a Reserved Event event (0x403b) 0b0 The common event is not implemented, or not counted. | |
| [26] | IDhi26 | IDhi26 corresponds to a Reserved Event event (0x403a) 0b0 The common event is not implemented, or not counted. | |
| [25] | IDhi25 | IDhi25 corresponds to a Reserved Event event (0x4039) 0b0 The common event is not implemented, or not counted. | |
| [24] | IDhi24 | IDhi24 corresponds to a Reserved Event event (0x4038) 0b0 The common event is not implemented, or not counted. | |

| Bits | Name | Description | Reset |
|------|--------|---|-------|
| [23] | IDHi23 | IDHi23 corresponds to a Reserved Event event (0x4037) 0b0 The common event is not implemented, or not counted. | |
| [22] | IDHi22 | IDHi22 corresponds to a Reserved Event event (0x4036) 0b0 The common event is not implemented, or not counted. | |
| [21] | IDHi21 | IDHi21 corresponds to a Reserved Event event (0x4035) 0b0 The common event is not implemented, or not counted. | |
| [20] | IDHi20 | IDHi20 corresponds to a Reserved Event event (0x4034) 0b0 The common event is not implemented, or not counted. | |
| [19] | IDHi19 | IDHi19 corresponds to a Reserved Event event (0x4033) 0b0 The common event is not implemented, or not counted. | |
| [18] | IDHi18 | IDHi18 corresponds to a Reserved Event event (0x4032) 0b0 The common event is not implemented, or not counted. | |
| [17] | IDHi17 | IDHi17 corresponds to a Reserved Event event (0x4031) 0b0 The common event is not implemented, or not counted. | |
| [16] | IDHi16 | IDHi16 corresponds to a Reserved Event event (0x4030) 0b0 The common event is not implemented, or not counted. | |
| [15] | IDHi15 | IDHi15 corresponds to a Reserved Event event (0x402f) 0b0 The common event is not implemented, or not counted. | |
| [14] | IDHi14 | IDHi14 corresponds to a Reserved Event event (0x402e) 0b0 The common event is not implemented, or not counted. | |
| [13] | IDHi13 | IDHi13 corresponds to a Reserved Event event (0x402d) 0b0 The common event is not implemented, or not counted. | |
| [12] | IDHi12 | IDHi12 corresponds to a Reserved Event event (0x402c) 0b0 The common event is not implemented, or not counted. | |
| [11] | IDHi11 | IDHi11 corresponds to a Reserved Event event (0x402b) 0b0 The common event is not implemented, or not counted. | |
| [10] | IDHi10 | IDHi10 corresponds to a Reserved Event event (0x402a) 0b0 The common event is not implemented, or not counted. | |

| Bits | Name | Description | Reset |
|------|-------|--|-------|
| [9] | IDhi9 | IDhi9 corresponds to a Reserved Event event (0x4029) 0b0 The common event is not implemented, or not counted. | |
| [8] | IDhi8 | IDhi8 corresponds to a Reserved Event event (0x4028) 0b0 The common event is not implemented, or not counted. | |
| [7] | IDhi7 | IDhi7 corresponds to a Reserved Event event (0x4027) 0b0 The common event is not implemented, or not counted. | |
| [6] | IDhi6 | IDhi6 corresponds to common event (0x4026) MEM_ACCESS_CHECKED_WR 0b1 The common event is implemented. | |
| [5] | IDhi5 | IDhi5 corresponds to common event (0x4025) MEM_ACCESS_CHECKED_RD 0b1 The common event is implemented. | |
| [4] | IDhi4 | IDhi4 corresponds to common event (0x4024) MEM_ACCESS_CHECKED 0b1 The common event is implemented. | |
| [3] | IDhi3 | IDhi3 corresponds to common event (0x4023) Reserved 0b0 The common event is not implemented, or not counted. | |
| [2] | IDhi2 | IDhi2 corresponds to common event (0x4022) ST_ALIGN_LAT 0b1 The common event is implemented. | |
| [1] | IDhi1 | IDhi1 corresponds to common event (0x4021) LD_ALIGN_LAT 0b1 The common event is implemented. | |
| [0] | IDhi0 | IDhi0 corresponds to common event (0x4020) LDST_ALIGN_LAT 0b1 The common event is implemented. | |

B.3.19 PMMIR, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE40

Reset value

See individual bit resets

Bit descriptions**Figure B-41: ext_pmmir bit assignments****Table B-44: PMMIR bit descriptions**

| Bits | Name | Description | Reset |
|--------|-------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | SLOTS | Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is implemented, this field must not be zero. | |

B.3.20 PMDEVARCH, Performance Monitors Device Architecture register

Identifies the programmers' model architecture of the Performance Monitor component.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes**Width**

32

Component

PMU

Register offset

0xFBC

Reset value

See individual bit resets

Bit descriptions

Figure B-42: ext_pmdevarch bit assignments

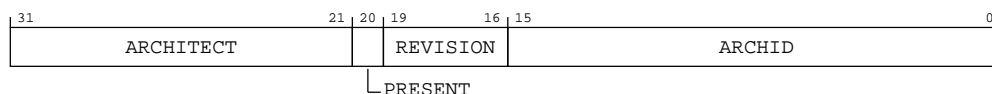


Table B-45: PMDEVARCH bit descriptions

| Bits | Name | Description | Reset |
|---------|-----------|---|-------|
| [31:21] | ARCHITECT | <p>Defines the architecture of the component. For Performance Monitors, this is Arm Limited.</p> <p>Bits [31:28] are the JEP106 continuation code, 0x4.</p> <p>Bits [27:21] are the JEP106 ID code, 0x3B.</p> | |
| [20] | PRESENT | <p>When set to 1, indicates that the DEVARCH is present.</p> <p>This field is 1 in Armv8.</p> | |
| [19:16] | REVISION | <p>Defines the architecture revision. For architectures defined by Arm this is the minor revision.</p> <p>For Performance Monitors, the revision defined by Armv8 is 0x0.</p> <p>All other values are reserved.</p> | |
| [15:0] | ARCHID | <p>Defines this part to be an Armv8 debug component. For architectures defined by Arm this is further subdivided.</p> <p>For Performance Monitors:</p> <ul style="list-style-type: none"> Bits [15:12] are the architecture version, 0x2. Bits [11:0] are the architecture part number, 0xA16. <p>This corresponds to Performance Monitors architecture version PMUv3.</p> <p>Note: The PMUv3 memory-mapped programmers' model can be used by devices other than Armv8 processors. Software must determine whether the PMU is attached to an Armv8 processor by using the ext-PMDEVAFF0 and ext-PMDEVAFF1 registers to discover the affinity of the PMU to any Armv8 processors.</p> | |

B.3.21 PMDEVID, Performance Monitors Device ID register

Provides information about features of the Performance Monitors implementation.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required from Armv8.2 and in any implementation that includes Armv8.2-PCSample. Otherwise, its location is RES0.



Before Armv8.2, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of ext-EDDEVID.PCSample.

Attributes

Width

32

Component

PMU

Register offset

0xFC8

Reset value

See individual bit resets

Bit descriptions

Figure B-43: ext_pmdevid bit assignments



Table B-46: PMDEVID bit descriptions

| Bits | Name | Description | Reset |
|--------|----------|---|-------|
| [31:4] | RES0 | Reserved | 0x0 |
| [3:0] | PCSample | Indicates the level of PC Sample-based Profiling support using Performance Monitors registers. 0b0001 PC Sample-based Profiling Extension is implemented in the Performance Monitors register space. | |

B.3.22 PMDEVTYPE, Performance Monitors Device Type register

Indicates to a debugger that this component is part of a PE's performance monitor interface.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFCC

Reset value

See individual bit resets

Bit descriptions**Figure B-44: ext_pmdevtype bit assignments****Table B-47: PMDEVTYPE bit descriptions**

| Bits | Name | Description | Reset |
|--------|-------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | SUB | Subtype. Must read as 0x1 to indicate this is a component within a PE. | |
| [3:0] | MAJOR | Major type. Must read as 0x6 to indicate this is a performance monitor component. | |

B.3.23 PMPIDR4, Performance Monitors Peripheral Identification Register 4

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes**Width**

32

Component

PMU

Register offset

0xFD0

Reset value

See individual bit resets

Bit descriptions

Figure B-45: ext_pmpidr4 bit assignments

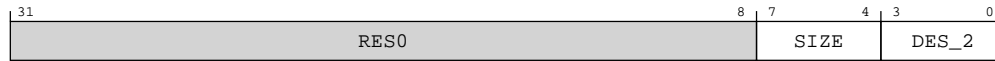


Table B-48: PMPIDR4 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | SIZE | 4KB count. 0b0000 The component uses a single 4KB block. | |
| [3:0] | DES_2 | Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. 0b0100 Arm Limited. This is bits[3:0] of the JEP106 continuation code. | |

B.3.24 PMPIDR0, Performance Monitors Peripheral Identification Register 0

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE0

Reset value

See individual bit resets

Bit descriptions

Figure B-46: ext_pmpidr0 bit assignments

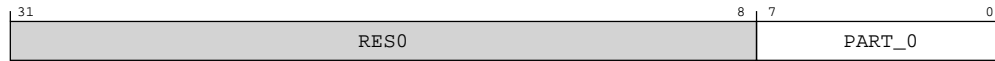


Table B-49: PMPIDR0 bit descriptions

| Bits | Name | Description | Reset |
|--------|--------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PART_0 | Part number, least significant byte. 0b01001111 Least significant byte of the PMU unit part. | |

B.3.25 PMPIDR1, Performance Monitors Peripheral Identification Register 1

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE4

Reset value

See individual bit resets

Bit descriptions

Figure B-47: ext_pmpidr1 bit assignments



Table B-50: PMPIDR1 bit descriptions

| Bits | Name | Description | Reset |
|--------|--------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | DES_0 | Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is 0b1011. 0b1011 Arm Limited. This is the least significant nibble of JEP106 ID code. | |
| [3:0] | PART_1 | Part number, most significant nibble. 0b1101 Part number, most significant nibble. | |

B.3.26 PMPIDR2, Performance Monitors Peripheral Identification Register 2

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE8

Reset value

See individual bit resets

Bit descriptions

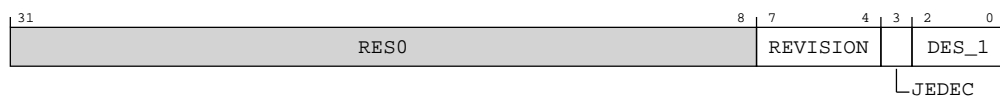
Figure B-48: ext_pmpidr2 bit assignments

Table B-51: PMPIDR2 bit descriptions

| Bits | Name | Description | Reset |
|--------|----------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | REVISION | Part major revision. Parts can also use this field to extend Part number to 16-bits. 0b0001 rOp1 | |
| [3] | JEDEC | RAO. Indicates a JEP106 identity code is used. 0b1 RES1. Indicates a JEP106 identity code is used | |
| [2:0] | DES_1 | Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. 0b011 Arm Limited. This is bits[6:4] of the JEP106 ID code. | |

B.3.27 PMPIDR3, Performance Monitors Peripheral Identification Register 3

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFEC

Reset value

See individual bit resets

Bit descriptions

Figure B-49: ext_pmpidr3 bit assignments

Table B-52: PMPIDR3 bit descriptions

| Bits | Name | Description | Reset |
|--------|--------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | REVAND | Part minor revision. Parts using ext-PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. 0b0000 | |
| [3:0] | CMOD | Customer modified. Indicates someone other than the Designer has modified the component. 0b0000 The component is not modified from the original design. | |

B.3.28 PMCIDR0, Performance Monitors Component Identification Register 0

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFF0

Reset value

See individual bit resets

Bit descriptions

Figure B-50: ext_pmcidr0 bit assignments

Table B-53: PMCIDR0 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_0 | Preamble. Must read as 0x0D. 0b00001101 Preamble byte 0 | |

B.3.29 PMCIDR1, Performance Monitors Component Identification Register 1

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFF4

Reset value

See individual bit resets

Bit descriptions

Figure B-51: ext_pmcidr1 bit assignments**Table B-54: PMCIDR1 bit descriptions**

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|---------|---|-------|
| [7:4] | CLASS | Component class. Reads as 0x9, debug component. 0b1001 Debug Component | |
| [3:0] | PRMBL_1 | Preamble. RAZ. 0b0000 Preamble | |

B.3.30 PMCIDR2, Performance Monitors Component Identification Register 2

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFF8

Reset value

See individual bit resets

Bit descriptions

Figure B-52: ext_pmcidr2 bit assignments



Table B-55: PMCIDR2 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|---------|---|-------|
| [7:0] | PRMBL_2 | Preamble. Must read as 0x05. 0b00000101 Preamble byte 2. | |

B.3.31 PMCIDR3, Performance Monitors Component Identification Register 3

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFFC

Reset value

See individual bit resets

Bit descriptions

Figure B-53: ext_pmcidr3 bit assignments



Table B-56: PMCIDR3 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_3 | Preamble. Must read as 0xB1. 0b10110001 Preamble byte 3. | |

B.4 External Debug register summary

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

Table B-57: External Debug register summary

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

B.4.1 EDRCR, External Debug Reserve Control Register

This register is used to allow imprecise entry to Debug state and clear sticky bits in ext-EDSCR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

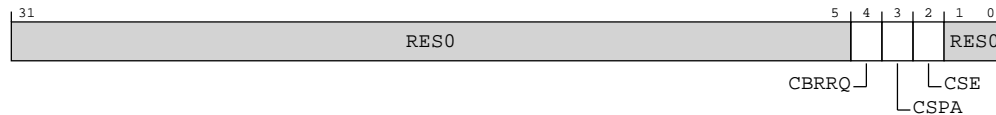
Debug

Register offset

0x090

Reset value

See individual bit resets

Bit descriptions**Figure B-54: ext_edcr bit assignments****Table B-58: EDRCR bit descriptions**

| Bits | Name | Description | Reset |
|--------|-------|---|-------|
| [31:5] | RES0 | Reserved | 0x0 |
| [4] | CBRRQ | This feature is not supported. Writes to this bit are ignored 0b0 No action. | |
| [3] | CSPA | Clear Sticky Pipeline Advance. This bit is used to clear the ext-EDSCR.PipeAdv bit to 0. 0b0 No action. 0b1 Clear the ext-EDSCR.PipeAdv bit to 0. | |
| [2] | CSE | Clear Sticky Error. Used to clear the ext-EDSCR cumulative error bits to 0. 0b0 No action. 0b1 Clear the ext-EDSCR.{TXU, RXO, ERR} bits, and, if the PE is in Debug state, the ext-EDSCR.ITO bit, to 0. | |
| [1:0] | RES0 | Reserved | 0b00 |

B.4.2 EDACR, External Debug Auxiliary Control Register

Allows implementations to support **IMPLEMENTATION DEFINED** controls.

Configurations

Changing this register from its reset value causes **IMPLEMENTATION DEFINED** behavior, including possible deviation from the architecturally-defined behavior.

If the EDACR contains any control bits that must be preserved over power down, then these bits must be accessible by the external debug interface when the OS Lock is locked, AArch64-OSLSR_EL1.OSLK == 1, and when the Core is powered off.

Attributes**Width**

32

Component

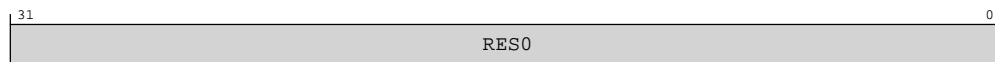
Debug

Register offset

0x094

Reset value

0x0

Bit descriptions**Figure B-55: ext_edacr bit assignments****Table B-59: EDACR bit descriptions**

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.4.3 EDPRCR, External Debug Power/Reset Control Register

Controls the PE functionality related to powerup, reset, and powerdown.

Configurations

If Armv8.3-DoPD is implemented then all fields in this register are in the Core power domain.

CORENPDRQ is the only field that is mapped between the EDPRCR and DBGPRCR and DBGPRCR_EL1.

Attributes**Width**

32

Component

Debug

Register offset

0x310

Reset value

See individual bit resets

Bit descriptions

Figure B-56: ext_edprcr bit assignments

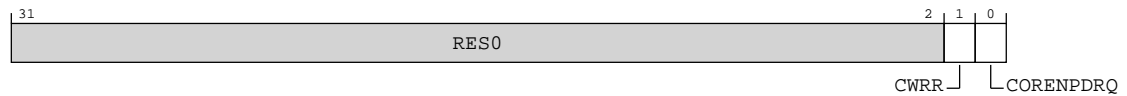


Table B-60: EDPRCR bit descriptions

| Bits | Name | Description | Reset |
|--------|-----------|--|-------|
| [31:2] | RES0 | Reserved | 0x0 |
| [1] | CWRR | This feature is not supported. Writes to this bit are ignored 0b0 No action. | |
| [0] | CORENPDRQ | This field is in the Core power domain, and permitted accesses to this field map to the AArch64-DBGPRCR_EL1.CORENPDRQ field. 0b0 If the system responds to a powerdown request, it powers down Core power domain. 0b1 If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain. | |

B.4.4 MIDR_EL1, Main ID Register

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

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xD00

Reset value

See individual bit resets

Bit descriptions

Figure B-57: ext_midr_el1 bit assignments

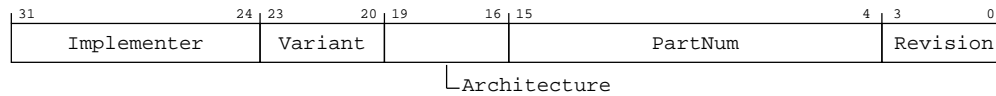


Table B-61: MIDR_EL1 bit descriptions

| Bits | Name | Description | Reset |
|---------|--------------|--|-------|
| [31:24] | Implementer | Indicates the implementer code. This value is: 0b01000001 Arm Limited | |
| [23:20] | Variant | An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product. 0b0000 rOp1 | |
| [19:16] | Architecture | Indicates the architecture code. This value is: 0b1111 Architecture is defined by ID registers | |
| [15:4] | PartNum | An IMPLEMENTATION DEFINED primary part number for the device. On processors implemented by Arm, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently. 0b110101001111 Neoverse™ V2 | |
| [3:0] | Revision | An IMPLEMENTATION DEFINED revision number for the device. 0b0001 rOp1 | |

B.4.5 EDPFR, External Debug Processor Feature Register

Provides information about implemented PE features.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

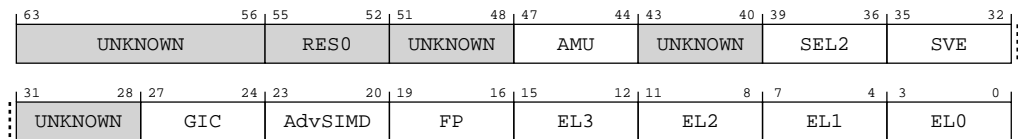
Debug

Register offset

0xD20

Reset value

See individual bit resets

Bit descriptions**Figure B-58: ext_edpfr bit assignments****Table B-62: EDPFR bit descriptions**

| Bits | Name | Description | Reset |
|---------|---------|--|--------|
| [63:56] | UNKNOWN | Reserved | |
| [55:52] | RES0 | Reserved | 0b0000 |
| [51:48] | UNKNOWN | Reserved | |
| [47:44] | AMU | Activity Monitors Extension. This value is : 0b0001 Activity Monitors Extension version 1 is implemented. | |
| [43:40] | UNKNOWN | Reserved | |
| [39:36] | SEL2 | Secure EL2. This value is : 0b0001 Secure EL2 is implemented. | |
| [35:32] | SVE | Scalable Vector Extension. This value is : 0b0001 SVE is implemented. | |
| [31:28] | UNKNOWN | Reserved | |
| [27:24] | GIC | System register GIC interface support. This field reads as 0x0 when GIC is disabled. 0b0011 System register interface to version 4.1 of the GIC CPU interface is supported. | |
| [23:20] | AdvSIMD | Advanced SIMD. This value is: 0b0001 As for 0b0000, and also includes support for half-precision floating-point arithmetic. | |
| [19:16] | FP | Floating Point. This value is: 0b0001 As for 0b0000, and also includes support for half-precision floating-point arithmetic. | |

| Bits | Name | Description | Reset |
|---------|------|---|-------|
| [15:12] | EL3 | AArch64 EL3 Exception level handling 0b0001 EL3 can be executed in AArch64 state only. | |
| [11:8] | EL2 | AArch64 EL2 Exception level handling 0b0001 EL2 can be executed in AArch64 state only. | |
| [7:4] | EL1 | AArch64 EL1 Exception level handling 0b0001 EL1 can be executed in AArch64 state only. | |
| [3:0] | EL0 | AArch64 EL0 Exception level handling 0b0001 EL0 can be executed in AArch64 state only. | |

B.4.6 EDDFR, External Debug Feature Register

Provides top-level information about the debug system.

Debuggers must use ext-EDDEVARCH to determine the Debug architecture version. For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offset

0xD28

Reset value

See individual bit resets

Bit descriptions

Figure B-59: ext_eddfr bit assignments

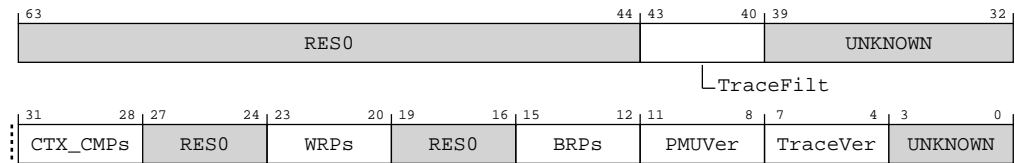


Table B-63: EDDFR bit descriptions

| Bits | Name | Description | Reset |
|---------|-----------|---|--------|
| [63:44] | RES0 | Reserved | 0x0 |
| [43:40] | TraceFilt | Armv8.4 Self-hosted Trace Extension version. This value is : 0b0001 Armv8.4 Self-hosted Trace Extension is implemented. | |
| [39:32] | UNKNOWN | Reserved | |
| [31:28] | CTX_CMPs | Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. In an Armv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of AArch64-ID_AA64DFR0_EL1.CTX_CMPs. | |
| [27:24] | RES0 | Reserved | 0b0000 |
| [23:20] | WRPs | Number of watchpoints, minus 1. The value of 0b0000 is reserved. In an Armv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of AArch64-ID_AA64DFR0_EL1.WRPs. | |
| [19:16] | RES0 | Reserved | 0b0000 |
| [15:12] | BRPs | Number of breakpoints, minus 1. The value of 0b0000 is reserved. In an Armv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of AArch64-ID_AA64DFR0_EL1.BRPs. | |
| [11:8] | PMUVer | Performance Monitors Extension version. | |
| [7:4] | TraceVer | Trace support. Indicates whether System register interface to a PE trace unit is implemented. 0b0001 PE trace unit System registers implemented. | |
| [3:0] | UNKNOWN | Reserved | |

B.4.7 EDDEVARCH, External Debug Device Architecture register

Identifies the programmers' model architecture of the external debug component.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes**Width**

32

Component

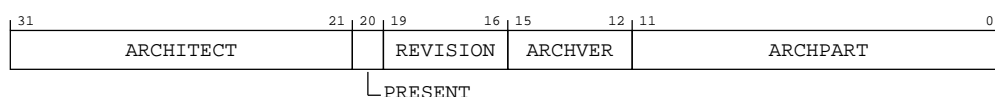
Debug

Register offset

0xFBC

Reset value

See individual bit resets

Bit descriptions**Figure B-60: ext_eddevarch bit assignments****Table B-64: EDDEVARCH bit descriptions**

| Bits | Name | Description | Reset |
|---------|-----------|--|-------|
| [31:21] | ARCHITECT | Defines the architecture of the component. For debug, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, 0x4. Bits [27:21] are the JEP106 ID code, 0x3B. | |
| [20] | PRESENT | When set to 1, indicates that the DEVARCH is present. This field is 1 in Armv8. | |
| [19:16] | REVISION | Defines the architecture revision. For architectures defined by Arm this is the minor revision. For debug, the revision defined by Armv8-A is 0x0. All other values are reserved. | |
| [15:12] | ARCHVER | Defines the architecture version of the component. This is the same value as AArch64-ID_AA64DFR0_EL1.DebugVer and AArch32-DBGDIDR.Version. This value is : 0b1001 Armv8.4 Debug architecture. | |
| [11:0] | ARCHPART | The fields ARCHVER and ARCHPART together form the field ARCHID, so that ARCHPART is ARCHID[11:0]. 0b101000010101 The part number of the Armv8-A debug component. | |

B.4.8 EDDEVID2, External Debug Device ID register 2

Reserved for future descriptions of features of the debug implementation.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFC0

Reset value

0x0

Bit descriptions

Figure B-61: ext_eddevid2 bit assignments



Table B-65: EDDEVID2 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.4.9 EDDEVID1, External Debug Device ID register 1

Provides extra information for external debuggers about features of the debug implementation.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFC4

Reset value

See individual bit resets

Bit descriptions**Figure B-62: ext_eddevid1 bit assignments****Table B-66: EDDEVID1 bit descriptions**

| Bits | Name | Description | Reset |
|--------|------------|--|-------|
| [31:4] | RES0 | Reserved | 0x0 |
| [3:0] | PCSROffset | This field indicates the offset applied to PC samples returned by reads of ext-EDPCSR. 0b0000 ext-EDPCSR not implemented. | |

B.4.10 EDDEVID, External Debug Device ID register 0

Provides extra information for external debuggers about features of the debug implementation.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain.

If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes**Width**

32

Component

Debug

Register offset

0xFC8

Reset value

See individual bit resets

Bit descriptions

Figure B-63: ext_eddevid bit assignments

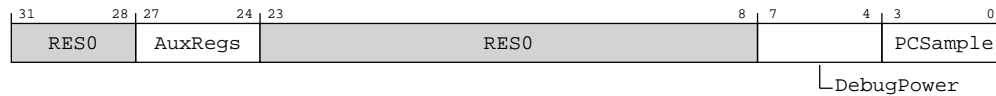


Table B-67: EDDEVID bit descriptions

| Bits | Name | Description | Reset |
|---------|------------|--|--------|
| [31:28] | RES0 | Reserved | 0b0000 |
| [27:24] | AuxRegs | Indicates support for Auxiliary registers. 0b0000 None supported. | |
| [23:8] | RES0 | Reserved | 0x0 |
| [7:4] | DebugPower | Indicates support for the Armv8.3-DoPD feature. 0b0001 Armv8.3-DoPD implemented. All registers in the external debug interface register map are implemented in the Core power domain. | |
| [3:0] | PCSample | Indicates the level of PC Sample-based Profiling support using external debug registers. 0b0000 PC Sample-based Profiling Extension is not implemented in the external debug registers space. | |

B.4.11 EDDEVTYPE, External Debug Device Type register

Indicates to a debugger that this component is part of a PE's debug logic.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFCC

Reset value

See individual bit resets

Bit descriptions

Figure B-64: ext_eddevtype bit assignments



Table B-68: EDDEVTYPE bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | SUB | Subtype. Must read as 0x1 to indicate this is a component within a PE. | |
| [3:0] | MAJOR | Major type. Must read as 0x5 to indicate this is a debug logic component. | |

B.4.12 EDPIDR4, External Debug Peripheral Identification Register 4

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFD0

Reset value

See individual bit resets

Bit descriptions

Figure B-65: ext_edpidr4 bit assignments



Table B-69: EDPIDR4 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | SIZE | 4KB count. 0b0000 The component uses a single 4KB block. | |
| [3:0] | DES_2 | Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. 0b0100 Arm Limited. This is bits[3:0] of the JEP106 continuation code. | |

B.4.13 EDPIDR0, External Debug Peripheral Identification Register 0

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE0

Reset value

See individual bit resets

Bit descriptions

Figure B-66: ext_edpidr0 bit assignments**Table B-70: EDPIDR0 bit descriptions**

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|--------|--|-------|
| [7:0] | PART_0 | Part number, least significant byte. 0b01001111 Least Significant byte of the debug part number | |

B.4.14 EDPIDR1, External Debug Peripheral Identification Register 1

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE4

Reset value

See individual bit resets

Bit descriptions

Figure B-67: ext_edpidr1 bit assignments



Table B-71: EDPIDR1 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | DES_0 | Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is 0b1011. 0b1011 Arm Limited. This is the least significant nibble of JEP106 ID code. | |

| Bits | Name | Description | Reset |
|-------|--------|---|-------|
| [3:0] | PART_1 | Part number, most significant nibble. 0b1101 Part number, most significant nibble. | |

B.4.15 EDPIDR2, External Debug Peripheral Identification Register 2

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE8

Reset value

See individual bit resets

Bit descriptions

Figure B-68: ext_edpidr2 bit assignments

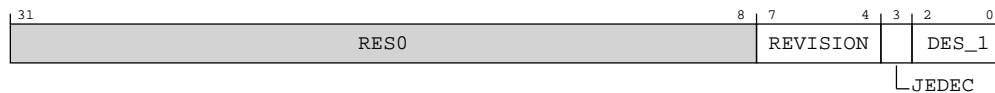


Table B-72: EDPIDR2 bit descriptions

| Bits | Name | Description | Reset |
|--------|----------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | REVISION | Part major revision. Parts can also use this field to extend Part number to 16-bits. 0b0001 rOp1 | |

| Bits | Name | Description | Reset |
|-------|-------|---|-------|
| [3] | JEDEC | RAO. Indicates a JEP106 identity code is used. 0b1 RAO. Indicates a JEP106 identity code is used | |
| [2:0] | DES_1 | Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. 0b011 Arm Limited. This is bits[6:4] of the JEP106 ID code. | |

B.4.16 EDPIDR3, External Debug Peripheral Identification Register 3

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFEC

Reset value

See individual bit resets

Bit descriptions

Figure B-69: ext_edpidr3 bit assignments



Table B-73: EDPIDR3 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|--------|--|-------|
| [7:4] | REVAND | Part minor revision. Parts using ext-EDPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. 0b0000 | |
| [3:0] | CMOD | Customer modified. Indicates someone other than the Designer has modified the component. 0b0000 The component is not modified from the original design. | |

B.4.17 EDCIDR0, External Debug Component Identification Register 0

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFF0

Reset value

See individual bit resets

Bit descriptions

Figure B-70: ext_edcidr0 bit assignments



Table B-74: EDCIDR0 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|---------|-------------|-------|
| [7:0] | PRMBL_0 | Preamble. | 0xD |

B.4.18 EDCIDR1, External Debug Component Identification Register 1

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFF4

Reset value

See individual bit resets

Bit descriptions

Figure B-71: ext_edcldr1 bit assignments



Table B-75: EDCIDR1 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|-----------------------------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | CLASS | Component class. Debug component. | 0x9 |
| [3:0] | PRMBL_1 | Preamble. | 0x0 |

B.4.19 EDCIDR2, External Debug Component Identification Register 2

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFF8

Reset value

See individual bit resets

Bit descriptions

Figure B-72: ext_edcidr2 bit assignments

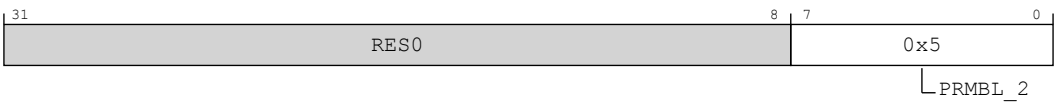


Table B-76: EDCIDR2 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|-------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_2 | Preamble. | 0x5 |

B.4.20 EDCIDR3, External Debug Component Identification Register 3

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

Configurations

If Armv8.3-DoPD is implemented, this register is in the Core power domain. If Armv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFFC

Reset value

See individual bit resets

Bit descriptions

Figure B-73: ext_edcldr3 bit assignments

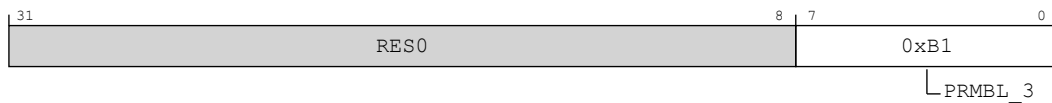


Table B-77: EDCIDR3 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|-------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_3 | Preamble. | 0xB1 |

B.5 External AMU register summary

The summary table provides an overview of all memory-mapped activity monitors registers in the core. Individual register descriptions provide detailed information.

Table B-78: External AMU register summary

| Offset | Name | Reset | Width | Description |
|--------|-----------------------------|---------------------------|--------|--|
| 0x400 | AMEVTYPER00 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 0 |
| 0x404 | AMEVTYPER01 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 0 |
| 0x408 | AMEVTYPER02 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 0 |
| 0x40C | AMEVTYPER03 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 0 |

| Offset | Name | Reset | Width | Description |
|--------|-----------------------------|---------------------------|--------|--|
| 0x480 | AMEVTYPER10 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 1 |
| 0x484 | AMEVTYPER11 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 1 |
| 0x488 | AMEVTYPER12 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 1 |
| 0x48C | AMEVTYPER13 | See individual bit resets | 32-bit | Activity Monitors Event Type Registers 1 |
| 0xCE0 | AMCGCR | See individual bit resets | 32-bit | Activity Monitors Counter Group Configuration Register |
| 0xE00 | AMCFGR | See individual bit resets | 32-bit | Activity Monitors Configuration Register |
| 0xE08 | AMIIDR | See individual bit resets | 32-bit | Activity Monitors Implementation Identification Register |
| 0xFBC | AMDEVARCH | See individual bit resets | 32-bit | Activity Monitors Device Architecture Register |
| 0xFCC | AMDEVTYPE | See individual bit resets | 32-bit | Activity Monitors Device Type Register |
| 0xFD0 | AMPIDR4 | See individual bit resets | 32-bit | Activity Monitors Peripheral Identification Register 4 |
| 0xFE0 | AMPIDR0 | See individual bit resets | 32-bit | Activity Monitors Peripheral Identification Register 0 |
| 0xFE4 | AMPIDR1 | See individual bit resets | 32-bit | Activity Monitors Peripheral Identification Register 1 |
| 0xFE8 | AMPIDR2 | See individual bit resets | 32-bit | Activity Monitors Peripheral Identification Register 2 |
| 0xFEC | AMPIDR3 | See individual bit resets | 32-bit | Activity Monitors Peripheral Identification Register 3 |
| 0xFF0 | AMCIDR0 | See individual bit resets | 32-bit | Activity Monitors Component Identification Register 0 |
| 0xFF4 | AMCIDR1 | See individual bit resets | 32-bit | Activity Monitors Component Identification Register 1 |
| 0xFF8 | AMCIDR2 | See individual bit resets | 32-bit | Activity Monitors Component Identification Register 2 |
| 0xFFC | AMCIDR3 | See individual bit resets | 32-bit | Activity Monitors Component Identification Register 3 |

B.5.1 AMEVTYPER00, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR00_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x400

Reset value

See individual bit resets

Bit descriptions

Figure B-74: ext_amevtyper00 bit assignments



Table B-79: AMEVTYPER00 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [31:25] | RAZ | Reserved | |
| [24:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: | |

B.5.2 AMEVTYPER01, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR01_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x404

Reset value

See individual bit resets

Bit descriptions

Figure B-75: ext_amevtyper01 bit assignments

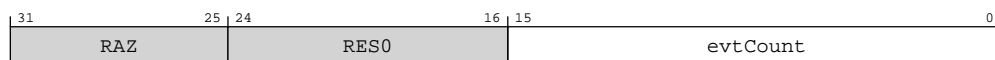


Table B-80: AMEVTYPER01 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [31:25] | RAZ | Reserved | |
| [24:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: | |

B.5.3 AMEVTYPER02, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR02_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x408

Reset value

See individual bit resets

Bit descriptions

Figure B-76: ext_amevtyper02 bit assignments**Table B-81: AMEVTYPER02 bit descriptions**

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [31:25] | RAZ | Reserved | |
| [24:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: | |

B.5.4 AMEVTYPER03, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR03_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x40C

Reset value

See individual bit resets

Bit descriptions

Figure B-77: ext_amevtyper03 bit assignments

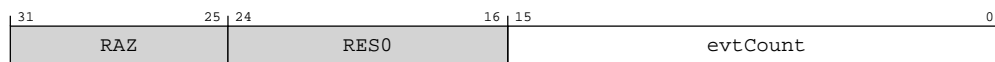


Table B-82: AMEVTYPER03 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [31:25] | RAZ | Reserved | |
| [24:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: | |

B.5.5 AMEVTYPER10, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR10_ELO counts.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

AMU

Register offset

0x480

Reset value

See individual bit resets

Bit descriptions**Figure B-78: ext_amevtyper10 bit assignments****Table B-83: AMEVTYPER10 bit descriptions**

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [31:25] | RAZ | Reserved | |
| [24:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | <p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>.</p> <p>It is IMPLEMENTATION DEFINED what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1<n>, then:</p> <ul style="list-style-type: none"> It is UNPREDICTABLE which event will be counted. The value read back is UNKNOWN. <p>Note: The event counted by ext-AMEVCNTR1<n> might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED. If the corresponding counter ext-AMEVCNTR1<n> is enabled, writes to this register have UNPREDICTABLE results.</p> | |

B.5.6 AMEVTYPER11, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR11_ELO counts.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

AMU

Register offset

0x484

Reset value

See individual bit resets

Bit descriptions**Figure B-79: ext_amevtyper11 bit assignments****Table B-84: AMEVTYPER11 bit descriptions**

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [31:25] | RAZ | Reserved | |
| [24:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | <p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>.</p> <p>It is IMPLEMENTATION DEFINED what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1<n>, then:</p> <ul style="list-style-type: none"> It is UNPREDICTABLE which event will be counted. The value read back is UNKNOWN. <p>Note: The event counted by ext-AMEVCNTR1<n> might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED. If the corresponding counter ext-AMEVCNTR1<n> is enabled, writes to this register have UNPREDICTABLE results.</p> | |

B.5.7 AMEVTYPER12, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR12_ELO counts.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

AMU

Register offset

0x488

Reset value

See individual bit resets

Bit descriptions**Figure B-80: ext_amevtyper12 bit assignments****Table B-85: AMEVTYPER12 bit descriptions**

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [31:25] | RAZ | Reserved | |
| [24:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | <p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>.</p> <p>It is IMPLEMENTATION DEFINED what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1<n>, then:</p> <ul style="list-style-type: none"> It is UNPREDICTABLE which event will be counted. The value read back is UNKNOWN. <p>Note: The event counted by ext-AMEVCNTR1<n> might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED. If the corresponding counter ext-AMEVCNTR1<n> is enabled, writes to this register have UNPREDICTABLE results.</p> | |

B.5.8 AMEVTYPER13, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR13_ELO counts.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

AMU

Register offset

0x48C

Reset value

See individual bit resets

Bit descriptions**Figure B-81: ext_amevtyper13 bit assignments****Table B-86: AMEVTYPER13 bit descriptions**

| Bits | Name | Description | Reset |
|---------|----------|---|-------|
| [31:25] | RAZ | Reserved | |
| [24:16] | RES0 | Reserved | 0x0 |
| [15:0] | evtCount | <p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>.</p> <p>It is IMPLEMENTATION DEFINED what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1<n>, then:</p> <ul style="list-style-type: none"> It is UNPREDICTABLE which event will be counted. The value read back is UNKNOWN. <p>Note: The event counted by ext-AMEVCNTR1<n> might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED. If the corresponding counter ext-AMEVCNTR1<n> is enabled, writes to this register have UNPREDICTABLE results.</p> | |

B.5.9 AMCGCR, Activity Monitors Counter Group Configuration Register

Provides information on the number of activity monitor event counters implemented within each counter group.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

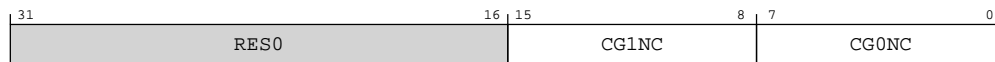
AMU

Register offset

0xCE0

Reset value

See individual bit resets

Bit descriptions**Figure B-82: ext_amcgcr bit assignments****Table B-87: AMCGCR bit descriptions**

| Bits | Name | Description | Reset |
|---------|-------|---|-------|
| [31:16] | RES0 | Reserved | 0x0 |
| [15:8] | CG1NC | Counter Group 1 Number of Counters. The number of counters in the auxiliary counter group. In AMUv1, the permitted range of values is 0 to 16. 0b00000011 Three counters in the auxiliary counter group | |
| [7:0] | CG0NC | Counter Group 0 Number of Counters. The number of counters in the architected counter group. In AMUv1, the value of this field is 4. 0b00000100 Four Counters in the architected counter group | |

B.5.10 AMCFGR, Activity Monitors Configuration Register

Global configuration register for the activity monitors.

Provides information on supported features, the number of counter groups implemented, the total number of activity monitor event counters implemented, and the size of the counters. AMCFGR is applicable to both the architected and the auxiliary counter groups.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

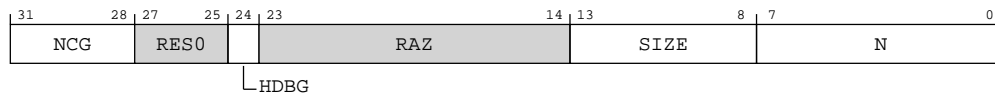
AMU

Register offset

0xE00

Reset value

See individual bit resets

Bit descriptions**Figure B-83: ext_amcfgr bit assignments****Table B-88: AMCFGR bit descriptions**

| Bits | Name | Description | Reset |
|---------|------|--|-------|
| [31:28] | NCG | Defines the number of counter groups. The following value is specified for this product. 0b0001 Two counter groups are implemented | |
| [27:25] | RES0 | Reserved | 0b000 |
| [24] | HDBG | Halt-on-debug supported. From Armv8, this feature must be supported, and so this bit is 0b1. 0b1 ext-AMCR.HDBG is read/write. | |
| [23:14] | RAZ | Reserved | |
| [13:8] | SIZE | Defines the size of activity monitor event counters. The size of the activity monitor event counters implemented by the Activity Monitors Extension is defined as [AMCFGR.SIZE + 1]. From Armv8, the counters are 64-bit, and so this field is 0b111111. Note: Software also uses this field to determine the spacing of counters in the memory-map. From Armv8, the counters are at doubleword-aligned addresses. 0b111111 64 bits. | |

| Bits | Name | Description | Reset |
|-------|------|--|-------|
| [7:0] | N | <p>Defines the number of activity monitor event counters.</p> <p>The total number of counters implemented in all groups by the Activity Monitors Extension is defined as [AMCFGR.N + 1].</p> <p>0b00000110</p> <p>Seven activity monitor event counters</p> | |

B.5.11 AMIIDR, Activity Monitors Implementation Identification Register

Defines the implementer and revisions of the AMU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xE08

Reset value

See individual bit resets

Bit descriptions

Figure B-84: ext_amiidr bit assignments

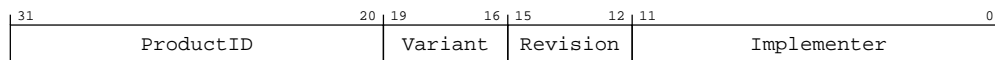


Table B-89: AMIIDR bit descriptions

| Bits | Name | Description | Reset |
|---------|-----------|---|-------|
| [31:20] | ProductID | <p>This field is an AMU part identifier.</p> <p>0b110101001111</p> <p>Neoverse™ V2</p> <p>If ext-AMPIDR0 is implemented, ext-AMPIDR0.PART_0 matches bits [27:20] of this field.</p> <p>If ext-AMPIDR1 is implemented, ext-AMPIDR1.PART_1 matches bits [31:28] of this field.</p> | |

| Bits | Name | Description | Reset |
|---------|-------------|--|-------|
| [19:16] | Variant | <p>This field distinguishes product variants or major revisions of the product.</p> <p>0b0000 rOp1</p> <p>If ext-AMPIDR2 is implemented, ext-AMPIDR2.REVISION matches AMIIDR.Variant.</p> | |
| [15:12] | Revision | <p>This field distinguishes minor revisions of the product.</p> <p>0b0001 rOp1</p> <p>If ext-AMPIDR3 is implemented, ext-AMPIDR3.REVAND matches AMIIDR.Revision.</p> | |
| [11:0] | Implementer | <p>Contains the JEP106 code of the company that implemented the AMU.</p> <p>For an Arm implementation, this field reads as 0x43B.</p> <p>Bits [11:8] contain the JEP106 continuation code of the implementer.</p> <p>Bit 7 is RES0</p> <p>Bits [6:0] contain the JEP106 identity code of the implementer.</p> <p>If ext-AMPIDR4 is implemented, ext-AMPIDR4.DES_2 matches bits [11:8] of this field.</p> <p>If ext-AMPIDR2 is implemented, ext-AMPIDR2.DES_1 matches bits [6:4] of this field.</p> <p>If ext-AMPIDR1 is implemented, ext-AMPIDR1.DES_0 matches bits [3:0] of this field.</p> | |

B.5.12 AMDEVARCH, Activity Monitors Device Architecture Register

Identifies the programmers' model architecture of the AMU component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFBC

Reset value

See individual bit resets

Bit descriptions

Figure B-85: ext_amdevarch bit assignments

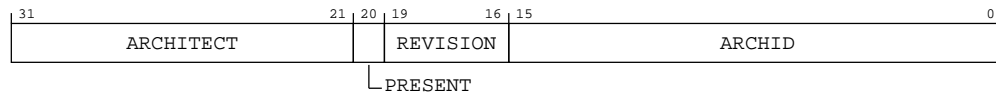


Table B-90: AMDEVARCH bit descriptions

| Bits | Name | Description | Reset |
|---------|-----------|--|-------|
| [31:21] | ARCHITECT | Defines the architecture of the component. For AMU, this is Arm Limited. | |
| [20] | PRESENT | When set to 1, indicates that the DEVARCH is present. 0b1 DEVARCH is present | |
| [19:16] | REVISION | Defines the architecture revision. For architectures defined by Arm this is the minor revision. 0b0000 Architecture revision is AMUv1. All other values are reserved. | |
| [15:0] | ARCHID | Defines this part to be an AMU component. For architectures defined by Arm this is further subdivided. For AMU: <ul style="list-style-type: none"> Bits [15:12] are the architecture version, 0x0. Bits [11:0] are the architecture part number, 0xA66. This corresponds to AMU architecture version AMUv1. | |

B.5.13 AMDEVTYPE, Activity Monitors Device Type Register

Indicates to a debugger that this component is part of a PE's performance monitor interface.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFCC

Reset value

See individual bit resets

Bit descriptions

Figure B-86: ext_amdevtype bit assignments

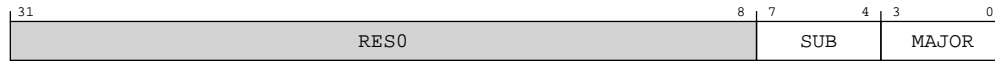


Table B-91: AMDEVTYPE bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | SUB | Subtype. Reads as 0x1, to indicate this is a component within a PE. | |
| [3:0] | MAJOR | Major type. Reads as 0x6, to indicate this is a performance monitor component. | |

B.5.14 AMPIDR4, Activity Monitors Peripheral Identification Register 4

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFD0

Reset value

See individual bit resets

Bit descriptions

Figure B-87: ext_ampidr4 bit assignments



Table B-92: AMPIDR4 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|-------|--|-------|
| [7:4] | SIZE | 4KB count. 0b0000 The component uses a single 4KB block. | |
| [3:0] | DES_2 | Designer. JEP106 continuation code, least significant nibble. The value of this field is IMPLEMENTATION DEFINED . For Arm Limited, this field is 0b0100. 0b0100 Arm Limited. This is bits[3:0] of the JEP106 continuation code. | |

B.5.15 AMPIDR0, Activity Monitors Peripheral Identification Register 0

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE0

Reset value

See individual bit resets

Bit descriptions

Figure B-88: ext_ampidr0 bit assignments



Table B-93: AMPIDR0 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|--------|---|-------|
| [7:0] | PART_0 | Part number, least significant byte. The value of this field is IMPLEMENTATION DEFINED . 0b01001111 Part number, least significant byte. | |

B.5.16 AMPIDR1, Activity Monitors Peripheral Identification Register 1

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE4

Reset value

See individual bit resets

Bit descriptions

Figure B-89: ext_ampidr1 bit assignments



Table B-94: AMPIDR1 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | DES_0 | Designer, least significant nibble of JEP106 ID code. The value of this field is IMPLEMENTATION DEFINED . For Arm Limited, this field is 0b1011. 0b1011 Designer, least significant nibble of JEP106 ID code. | |

| Bits | Name | Description | Reset |
|-------|--------|---|-------|
| [3:0] | PART_1 | Part number, most significant nibble. The value of this field is IMPLEMENTATION DEFINED . 0b1101 Part number, most significant nibble. | |

B.5.17 AMPIDR2, Activity Monitors Peripheral Identification Register 2

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE8

Reset value

See individual bit resets

Bit descriptions

Figure B-90: ext_ampidr2 bit assignments

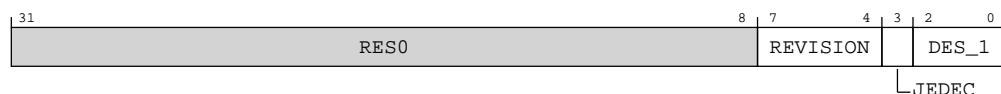


Table B-95: AMPIDR2 bit descriptions

| Bits | Name | Description | Reset |
|--------|----------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | REVISION | Part major revision. Parts can also use this field to extend Part number to 16-bits. The value of this field is IMPLEMENTATION DEFINED . 0b0001 rOp1 | |

| Bits | Name | Description | Reset |
|-------|-------|---|-------|
| [3] | JEDEC | RAO. Indicates a JEP106 identity code is used. 0b1 RAO. Indicates a JEP106 identity code is used. | |
| [2:0] | DES_1 | Designer, most significant bits of JEP106 ID code. The value of this field is IMPLEMENTATION DEFINED . For Arm Limited, this field is 0b011. 0b011 Arm Limited. This is bits[6:4] of the JEP106 ID code. | |

B.5.18 AMPIDR3, Activity Monitors Peripheral Identification Register 3

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFEC

Reset value

See individual bit resets

Bit descriptions

Figure B-91: ext_ampidr3 bit assignments



Table B-96: AMPIDR3 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|--------|--|-------|
| [7:4] | REVAND | <p>Part minor revision. Parts using ext-AMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number.</p> <p>The value of this field is IMPLEMENTATION DEFINED.</p> <p>0b0000</p> | |
| [3:0] | CMOD | <p>Customer modified. Indicates someone other than the Designer has modified the component.</p> <p>The value of this field is IMPLEMENTATION DEFINED.</p> <p>0b0000</p> <p>The component is not modified from the original design.</p> | |

B.5.19 AMCIDR0, Activity Monitors Component Identification Register 0

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFF0

Reset value

See individual bit resets

Bit descriptions

Figure B-92: ext_amcidr0 bit assignments



Table B-97: AMCIDR0 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:8] | RES0 | Reserved | 0x0 |

| Bits | Name | Description | Reset |
|-------|---------|---|-------|
| [7:0] | PRMBL_0 | Preamble. Must read as 0x0D. 0b00001101 Preamble | |

B.5.20 AMCIDR1, Activity Monitors Component Identification Register 1

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFF4

Reset value

See individual bit resets

Bit descriptions

Figure B-93: ext_amcidr1 bit assignments



Table B-98: AMCIDR1 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | CLASS | Component class. Reads as 0x9, CoreSight component. 0b1001 CoreSight component. | |
| [3:0] | PRMBL_1 | Preamble. Reads as 0x0. 0b0000 Preamble | |

B.5.21 AMCIDR2, Activity Monitors Component Identification Register 2

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFF8

Reset value

See individual bit resets

Bit descriptions

Figure B-94: ext_amcidr2 bit assignments



Table B-99: AMCIDR2 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_2 | Preamble. Reads as 0x05. 0b00000101 Preamble byte 2 | |

B.5.22 AMCIDR3, Activity Monitors Component Identification Register 3

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

AMU

Register offset

0xFFC

Reset value

See individual bit resets

Bit descriptions**Figure B-95: ext_amcidr3 bit assignments****Table B-100: AMCIDR3 bit descriptions**

| Bits | Name | Description | Reset |
|--------|---------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_3 | Preamble. Reads as 0xB1. 0b10110001 Preamble byte 3 | |

B.6 External ETE register summary

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

Table B-101: External ETE register summary

| Offset | Name | Reset | Width | Description |
|--------|------------|---------------------------|--------|----------------------------|
| 0x018 | TRCAUXCTLR | 0x0 | 32-bit | Auxillary Control Register |
| 0x180 | TRCIDR8 | See individual bit resets | 32-bit | ID Register 8 |
| 0x184 | TRCIDR9 | See individual bit resets | 32-bit | ID Register 9 |
| 0x188 | TRCIDR10 | See individual bit resets | 32-bit | ID Register 10 |
| 0x18C | TRCIDR11 | See individual bit resets | 32-bit | ID Register 11 |
| 0x190 | TRCIDR12 | 0x0 | 32-bit | ID Register 12 |
| 0x194 | TRCIDR13 | 0x0 | 32-bit | ID Register 13 |
| 0x1C0 | TRCIMSPEC0 | See individual bit resets | 32-bit | IMP DEF Register 0 |
| 0x1E0 | TRCIDR0 | See individual bit resets | 32-bit | ID Register 0 |
| 0x1E4 | TRCIDR1 | See individual bit resets | 32-bit | ID Register 1 |

| Offset | Name | Reset | Width | Description |
|--------|-------------|---------------------------|--------|--------------------------------------|
| 0x1E8 | TRCIDR2 | See individual bit resets | 32-bit | ID Register 2 |
| 0x1EC | TRCIDR3 | See individual bit resets | 32-bit | ID Register 3 |
| 0x1F0 | TRCIDR4 | See individual bit resets | 32-bit | ID Register 4 |
| 0x1F4 | TRCIDR5 | See individual bit resets | 32-bit | ID Register 5 |
| 0x1F8 | TRCIDR6 | 0x0 | 32-bit | ID Register 6 |
| 0x1FC | TRCIDR7 | 0x0 | 32-bit | ID Register 7 |
| 0xF00 | TRCITCTRL | See individual bit resets | 32-bit | Integration Mode Control Register |
| 0xFA0 | TRCCLAIMSET | See individual bit resets | 32-bit | Claim Tag Set Register |
| 0xFA4 | TRCCLAIMCLR | See individual bit resets | 32-bit | Claim Tag Clear Register |
| 0xFBC | TRCDEVARCH | See individual bit resets | 32-bit | Device Architecture Register |
| 0xFC0 | TRCDEVID2 | 0x0 | 32-bit | Device Configuration Register 2 |
| 0xFC4 | TRCDEVID1 | 0x0 | 32-bit | Device Configuration Register 1 |
| 0xFC8 | TRCDEVID | 0x0 | 32-bit | Device Configuration Register |
| 0xFCC | TRCDEVTYPE | See individual bit resets | 32-bit | Device Type Register |
| 0xFD0 | TRCPIDR4 | See individual bit resets | 32-bit | Peripheral Identification Register 4 |
| 0xFD4 | TRCPIDR5 | 0x0 | 32-bit | Peripheral Identification Register 5 |
| 0xFD8 | TRCPIDR6 | 0x0 | 32-bit | Peripheral Identification Register 6 |
| 0xFDC | TRCPIDR7 | 0x0 | 32-bit | Peripheral Identification Register 7 |
| 0xFE0 | TRCPIDR0 | See individual bit resets | 32-bit | Peripheral Identification Register 0 |
| 0xFE4 | TRCPIDR1 | See individual bit resets | 32-bit | Peripheral Identification Register 1 |
| 0xFE8 | TRCPIDR2 | See individual bit resets | 32-bit | Peripheral Identification Register 2 |
| 0xFEC | TRCPIDR3 | See individual bit resets | 32-bit | Peripheral Identification Register 3 |
| 0xFF0 | TRCCIDR0 | See individual bit resets | 32-bit | Component Identification Register 0 |
| 0xFF4 | TRCCIDR1 | See individual bit resets | 32-bit | Component Identification Register 1 |
| 0xFF8 | TRCCIDR2 | See individual bit resets | 32-bit | Component Identification Register 2 |
| 0xFFC | TRCCIDR3 | See individual bit resets | 32-bit | Component Identification Register 3 |

B.6.1 TRCAUXCTLR, Auxillary Control Register

The function of this register is **IMPLEMENTATION DEFINED**.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

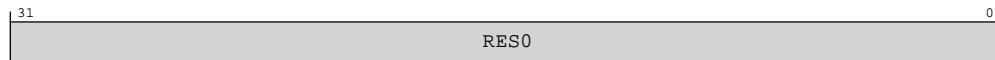
ETE

Register offset

0x018

Reset value

0x0

Bit descriptions**Figure B-96: ext_trcauxctlr bit assignments****Table B-102: TRCAUXCTLR bit descriptions**

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.6.2 TRCIDR8, ID Register 8

Returns the maximum speculation depth of the instruction trace element stream.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

ETE

Register offset

0x180

Reset value

See individual bit resets

Bit descriptions**Figure B-97: ext_trcidr8 bit assignments**

Table B-103: TRCIDR8 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|---|-------|
| [31:0] | MAXSPEC | Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of PO elements in the trace element stream that can be speculative at any time. | |

B.6.3 TRCIDR9, ID Register 9

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

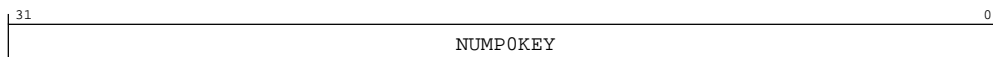
0x184

Reset value

See individual bit resets

Bit descriptions

Figure B-98: ext_trcidr9 bit assignments

**Table B-104: TRCIDR9 bit descriptions**

| Bits | Name | Description | Reset |
|--------|----------|--|-------|
| [31:0] | NUMPOKEY | Indicates the number of PO right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures. | |

B.6.4 TRCIDR10, ID Register 10

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

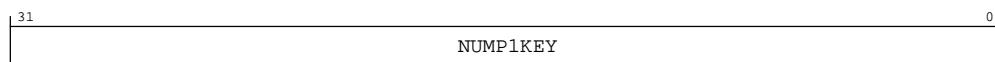
ETE

Register offset

0x188

Reset value

See individual bit resets

Bit descriptions**Figure B-99: ext_trcidr10 bit assignments****Table B-105: TRCIDR10 bit descriptions**

| Bits | Name | Description | Reset |
|--------|----------|--|-------|
| [31:0] | NUMP1KEY | Indicates the number of P1 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures. | |

B.6.5 TRCIDR11, ID Register 11

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

ETE

Register offset

0x18C

Reset value

See individual bit resets

Bit descriptions

Figure B-100: ext_trcidr11 bit assignments

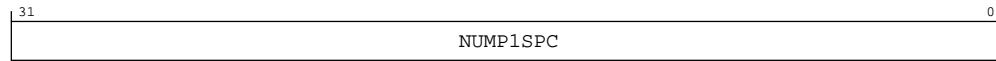


Table B-106: TRCIDR11 bit descriptions

| Bits | Name | Description | Reset |
|--------|----------|--|-------|
| [31:0] | NUMP1SPC | Indicates the number of special P1 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures. | |

B.6.6 TRCIDR12, ID Register 12

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x190

Reset value

0x0

Bit descriptions

Figure B-101: ext_trcidr12 bit assignments



Table B-107: TRCIDR12 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.6.7 TRCIDR13, ID Register 13

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x194

Reset value

0x0

Bit descriptions

Figure B-102: ext_trcidr13 bit assignments

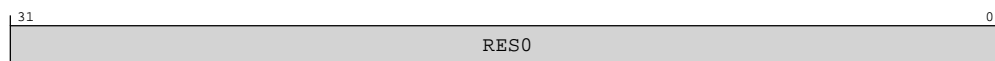


Table B-108: TRCIDR13 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.6.8 TRCIMSPEC0, IMP DEF Register 0

TRCIMSPEC0 shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

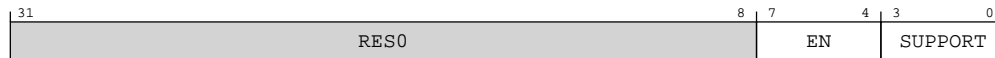
ETE

Register offset

0x1C0

Reset value

See individual bit resets

Bit descriptions**Figure B-103: ext_trcimspec0 bit assignments****Table B-109: TRCIMSPEC0 bit descriptions**

| Bits | Name | Description | Reset |
|--------|---------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | EN | Enable. Controls whether the IMPLEMENTATION DEFINED features are enabled. 0b0000 The IMPLEMENTATION DEFINED features are not enabled. The trace unit must behave as if the IMPLEMENTATION DEFINED features are not supported. | |
| [3:0] | SUPPORT | Indicates whether the implementation supports IMPLEMENTATION DEFINED features. 0b0000 No IMPLEMENTATION DEFINED features are supported. | |

B.6.9 TRCIDR0, ID Register 0

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

ETE

Register offset

0x1E0

Reset value

See individual bit resets

Bit descriptions

Figure B-104: ext_trcidr0 bit assignments

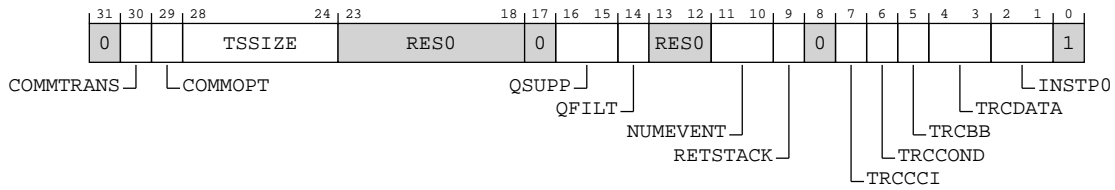


Table B-110: TRCIDR0 bit descriptions

| Bits | Name | Description | Reset |
|---------|-----------|---|-------|
| [31] | RES0 | Reserved | 0b0 |
| [30] | COMMTRANS | Transaction Start element behavior. 0b0 Transaction Start elements are PO elements. | |
| [29] | COMMOPT | Indicates the contents and encodings of Cycle count packets. 0b1 Commit mode 1. | |
| [28:24] | TSSIZE | Indicates that the trace unit implements Global timestamping and the size of the timestamp value. 0b01000 Global timestamping implemented with a 64-bit timestamp value. | |
| [23:17] | RES0 | Reserved | 0b0 |
| [16:15] | QSUPP | Indicates that the trace unit implements Q element support. 0b00 Q element support is not implemented. | |
| [14] | QFILT | Indicates if the trace unit implements Q element filtering. 0b0 Q element filtering is not implemented. | |
| [13:12] | RES0 | Reserved | 0b00 |
| [11:10] | NUMEVENT | Indicates the number of ETEEvents implemented. 0b11 The trace unit supports 4 ETEEvents. | |
| [9] | RETSTACK | Indicates if the trace unit supports the return stack. 0b1 Return stack implemented. | |
| [8] | RES0 | Reserved | 0b0 |
| [7] | TRCCCI | Indicates if the trace unit implements cycle counting. 0b1 Cycle counting implemented. | |

| Bits | Name | Description | Reset |
|-------|---------|---|-------|
| [6] | TRCCOND | Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. 0b0 Conditional instruction tracing not implemented. | |
| [5] | TRCBB | Indicates if the trace unit implements branch broadcasting. 0b1 Branch broadcasting implemented. | |
| [4:3] | TRCDATA | Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. 0b00 Data tracing not implemented. | |
| [2:1] | INSTPO | Indicates if load and store instructions are PO instructions. Load and store instructions as PO instructions is not implemented in ETE and this field is reserved for other trace architectures. 0b00 Load and store instructions are not PO instructions. | |
| [0] | RES1 | Reserved | 0b1 |

B.6.10 TRCIDR1, ID Register 1

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1E4

Reset value

See individual bit resets

Bit descriptions

Figure B-105: ext_trcidr1 bit assignments

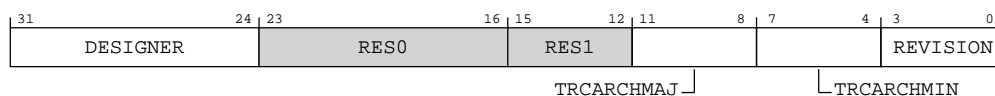


Table B-111: TRCIDR1 bit descriptions

| Bits | Name | Description | Reset |
|---------|------------|---|------------|
| [31:24] | DESIGNER | Indicates which company designed the trace unit. The permitted values of this field are the same as AArch64-MIDR_EL1.Implementer. 0b01000001 Arm Limited | |
| [23:16] | RES0 | Reserved | 0b00000000 |
| [15:12] | RES1 | Reserved | 0b0001 |
| [11:8] | TRCARCHMAJ | Major architecture version. 0b1111 If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH. | |
| [7:4] | TRCARCHMIN | Minor architecture version. 0b1111 If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH. | |
| [3:0] | REVISION | Implementation revision that identifies the revision of the trace and OS Lock registers. 0b0000 Revision 0 | |

B.6.11 TRCIDR2, ID Register 2

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1E8

Reset value

See individual bit resets

Bit descriptions

Figure B-106: ext_trcidr2 bit assignments

Table B-112: TRCIDR2 bit descriptions

| Bits | Name | Description | Reset |
|---------|----------|---|---------|
| [31] | WFXMODE | Indicates whether WFI and WFE instructions are classified as PO instructions. 0b1 WFI and WFE instructions are classified as PO instructions. | |
| [30:29] | VMIDOPT | Indicates the options for Virtual context identifier selection. 0b10 Virtual context identifier selection not supported. ext-TRCCONFIGR.VMIDOPT is RES1. | |
| [28:25] | CCSIZE | Indicates the size of the cycle counter. 0b0000 The cycle counter is 12 bits in length. | |
| [24:15] | RES0 | Reserved | 0b00000 |
| [14:10] | VMIDSIZE | Indicates the trace unit Virtual context identifier size. 0b00100 32-bit Virtual context identifier size. | |
| [9:5] | CIDSIZE | Indicates the Context identifier size. 0b00100 32-bit Context identifier size. | |
| [4:0] | IASIZE | Virtual instruction address size. 0b01000 Maximum of 64-bit instruction address size. | |

B.6.12 TRCIDR3, ID Register 3

Returns the base architecture of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1EC

Reset value

See individual bit resets

Bit descriptions

Figure B-107: ext_trcidr3 bit assignments

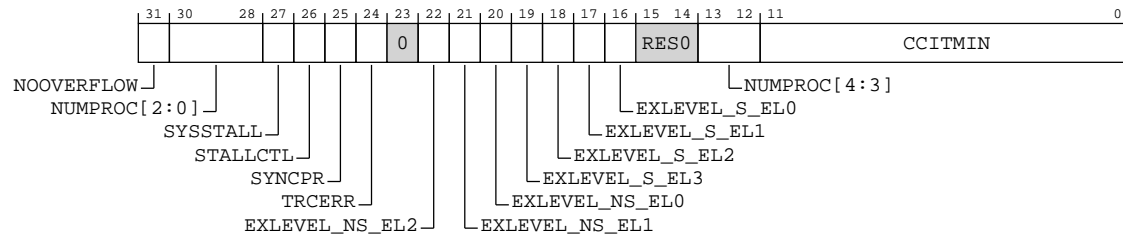


Table B-113: TRCIDR3 bit descriptions

| Bits | Name | Description | Reset |
|----------------|----------------|--|-------|
| [31] | NOOVERFLOW | Indicates if overflow prevention is implemented. 0b0 Overflow prevention is not implemented. | |
| [13:12, 30:28] | NUMPROC | Indicates the number of PEs available for tracing. 0b00000 The trace unit can trace one PE. | |
| [27] | SYSSTALL | Indicates if stalling of the PE is permitted. 0b0 Stalling of the PE is not permitted. | |
| [26] | STALLCTL | Indicates if trace unit implements stalling of the PE. 0b0 Stalling of the PE is not implemented. | |
| [25] | SYNCPR | Indicates if an implementation has a fixed synchronization period. 0b0 ext-TRCSYNCPR is read/write so software can change the synchronization period. | |
| [24] | TRCERR | Indicates forced tracing of System Error exceptions is implemented. 0b1 Forced tracing of System Error exceptions is implemented. | |
| [23] | RES0 | Reserved | 0b0 |
| [22] | EXLEVEL_NS_EL2 | Indicates if Non-secure EL2 implemented. 0b1 Non-secure EL2 is implemented. | |
| [21] | EXLEVEL_NS_EL1 | Indicates if Non-secure EL1 implemented. 0b1 Non-secure EL1 is implemented. | |
| [20] | EXLEVEL_NS_ELO | Indicates if Non-secure ELO implemented. 0b1 Non-secure ELO is implemented. | |

| Bits | Name | Description | Reset |
|---------|---------------|--|-------|
| [19] | EXLEVEL_S_EL3 | Indicates if Secure EL3 implemented. 0b1 Secure EL3 is implemented. | |
| [18] | EXLEVEL_S_EL2 | Indicates if Secure EL2 implemented. 0b1 Secure EL2 is implemented. | |
| [17] | EXLEVEL_S_EL1 | Indicates if Secure EL1 implemented. 0b1 Secure EL1 is implemented. | |
| [16] | EXLEVEL_S_ELO | Indicates if Secure ELO implemented. 0b1 Secure ELO is implemented. | |
| [15:14] | RES0 | Reserved | 0b00 |
| [11:0] | CCITMIN | Indicates the minimum value that can be programmed in ext-TRCCCCTLR.THRESHOLD. If ext-TRCIDR0.TRCCCI == 0b1 then the minimum value of this field is 0x001. If ext-TRCIDR0.TRCCCI == 0b0 then this field is zero. | |

B.6.13 TRCIDR4, ID Register 4

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1F0

Reset value

See individual bit resets

Bit descriptions

Figure B-108: ext_trcidr4 bit assignments

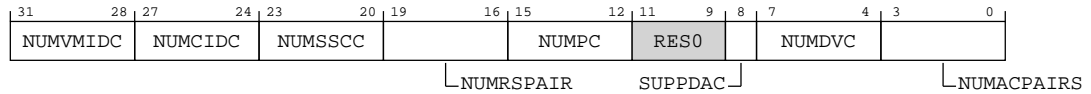


Table B-114: TRCIDR4 bit descriptions

| Bits | Name | Description | Reset |
|---------|------------|--|-------|
| [31:28] | NUMVMIDC | Indicates the number of Virtual Context Identifier Comparators that are available for tracing. 0b0001 The implementation has one Virtual Context Identifier Comparator. | |
| [27:24] | NUMCIDC | Indicates the number of Context Identifier Comparators that are available for tracing. 0b0001 The implementation has one Context Identifier Comparator. | |
| [23:20] | NUMSSCC | Indicates the number of Single-shot Comparator Controls that are available for tracing. 0b0001 The implementation has one Single-shot Comparator Control. | |
| [19:16] | NUMRSPAIR | Indicates the number of resource selector pairs that are available for tracing. 0b0111 The implementation has eight resource selector pairs. | |
| [15:12] | NUMPC | Indicates the number of PE Comparator Inputs that are available for tracing. 0b0000 No PE Comparator Inputs are available. | |
| [11:9] | RES0 | Reserved | 0b000 |
| [8] | SUPPDAC | Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. 0b0 Data address comparisons not implemented. | |
| [7:4] | NUMDVC | Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. 0b0000 No data value comparators implemented. | |
| [3:0] | NUMACPAIRS | Indicates the number of Address Comparator pairs that are available for tracing. 0b0100 The implementation has four Address Comparator pairs. | |

B.6.14 TRCIDR5, ID Register 5

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1F4

Reset value

See individual bit resets

Bit descriptions

Figure B-109: ext_trcidr5 bit assignments

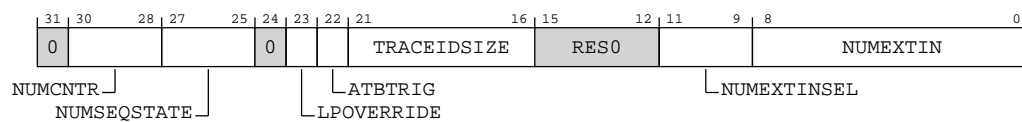


Table B-115: TRCIDR5 bit descriptions

| Bits | Name | Description | Reset |
|---------|-------------|--|-------|
| [31] | RES0 | Reserved | 0b0 |
| [30:28] | NUMCNTR | Indicates the number of Counters that are available for tracing. 0b010 Two Counters implemented. | |
| [27:25] | NUMSEQSTATE | Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. 0b100 Four Sequencer states are implemented. | |
| [24] | RES0 | Reserved | 0b0 |
| [23] | LPOVERRIDE | Indicates support for Low-power Override Mode. 0b0 The trace unit does not support Low-power Override Mode. | |
| [22] | ATBTRIG | Indicates if the implementation can support ATB triggers. 0b1 The implementation supports ATB triggers. | |

| Bits | Name | Description | Reset |
|---------|-------------|---|--------|
| [21:16] | TRACEIDSIZE | Indicates the trace ID width. 0b000111 The implementation supports a 7-bit trace ID. | |
| [15:12] | RES0 | Reserved | 0b0000 |
| [11:9] | NUMEXTINSEL | Indicates how many External Input Selector resources are implemented. 0b100 4 External Input Selector resources are available. | |
| [8:0] | NUMEXTIN | Indicates how many External Inputs are implemented. 0b11111111 Unified PMU event selection. All other values are reserved. | |

B.6.15 TRCIDR6, ID Register 6

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1F8

Reset value

0x0

Bit descriptions

Figure B-110: ext_trcidr6 bit assignments



Table B-116: TRCIDR6 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.6.16 TRCIDR7, ID Register 7

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1FC

Reset value

0x0

Bit descriptions

Figure B-111: ext_trcidr7 bit assignments



Table B-117: TRCIDR7 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.6.17 TRCITCTRL, Integration Mode Control Register

A component can use TRCITCTRL to dynamically switch between functional mode and integration mode. In integration mode, topology detection is enabled. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xF00

Reset value

See individual bit resets

Bit descriptions

Figure B-112: ext_trcitctrl bit assignments

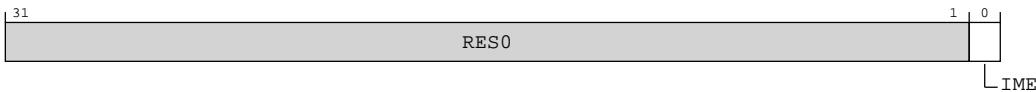


Table B-118: TRCITCTRL bit descriptions

| Bits | Name | Description | Reset |
|--------|------|--|-------|
| [31:1] | RES0 | Reserved | 0x0 |
| [0] | IME | Integration Mode Enable. 0b0 The component must enter functional mode. 0b1 The component must enter integration mode, and enable support for topology detection and integration testing. This bit is RES0 if no topology detection or integration functionality is implemented. | |

B.6.18 TRCCLAIMSET, Claim Tag Set Register

In conjunction with ext-TRCCLAIMCLR, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information see the CoreSight Architecture Specification.

Configurations

The number of claim tag bits implemented is IMPLEMENTATION DEFINED. Arm recommends that implementations support a minimum of four claim tag bits, that is, SET[3:0] reads as 0b1111.

Attributes**Width**

32

Component

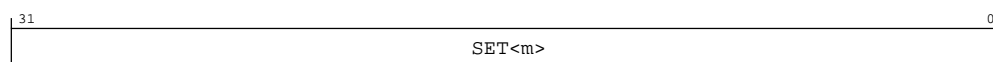
ETE

Register offset

0xFA0

Reset value

See individual bit resets

Bit descriptions**Figure B-113: ext_trclaimset bit assignments****Table B-119: TRCCLAIMSET bit descriptions**

| Bits | Name | Description | Reset |
|--------|--------|---|-------|
| [31:0] | SET<m> | <p>Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1.</p> <p>0b0</p> <p>On a read: Claim Tag bit m is not implemented.</p> <p>On a write: Ignored.</p> <p>0b1</p> <p>On a read: Claim Tag bit m is implemented.</p> <p>On a write: Set Claim Tag bit m to 0b1.</p> | |

B.6.19 TRCCLAIMCLR, Claim Tag Clear Register

In conjunction with ext-TRCCLAIMSET, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component
ETE

Register offset
0xFA4

Reset value
See individual bit resets

Bit descriptions

Figure B-114: ext_trclaimclr bit assignments

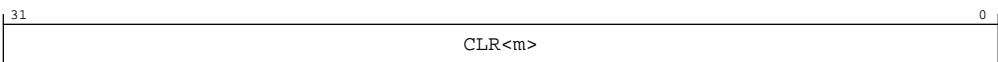


Table B-120: TRCCLAIMCLR bit descriptions

| Bits | Name | Description | Reset |
|--------|--------|---|-------|
| [31:0] | CLR<m> | <p>Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0.</p> <p>0b0</p> <p>On a read: Claim Tag bit m is not set.</p> <p>On a write: Ignored.</p> <p>0b1</p> <p>On a read: Claim Tag bit m is set.</p> <p>On a write: Clear Claim tag bit m to 0b0.</p> <p>The number of Claim Tag bits implemented is indicated in ext-TRCCLAIMSET.</p> | |

B.6.20 TRCDEVARCH, Device Architecture Register

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

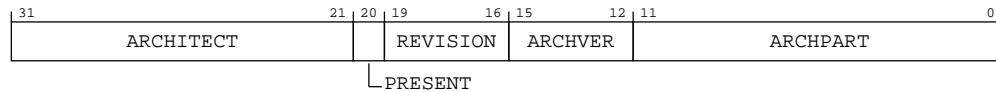
Component
ETE

Register offset

0xFBC

Reset value

See individual bit resets

Bit descriptions**Figure B-115: ext_trcdevarch bit assignments****Table B-121: TRCDEVARCH bit descriptions**

| Bits | Name | Description | Reset |
|---------|-----------|---|-------|
| [31:21] | ARCHITECT | Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code. 0b01000111011 JEP106 continuation code 0x4, ID code 0x3B. Arm Limited. | |
| [20] | PRESENT | DEVARCH Present. Defines that the DEVARCH register is present. 0b1 Device Architecture information present. | |
| [19:16] | REVISION | Revision. Defines the architecture revision of the component. 0b0000 ETE Version 1.0. | |
| [15:12] | ARCHVER | Architecture Version. Defines the architecture version of the component. 0b0101 ETE Version 1. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12]. This field reads as 0x5. | |
| [11:0] | ARCHPART | Architecture Part. Defines the architecture of the component. 0b101000010011 Arm PE trace architecture. | |

B.6.21 TRCDEVID2, Device Configuration Register 2

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

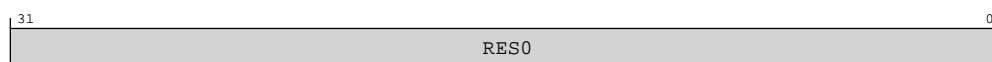
ETE

Register offset

0xFC0

Reset value

0x0

Bit descriptions**Figure B-116: ext_trcdevid2 bit assignments****Table B-122: TRCDEVID2 bit descriptions**

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.6.22 TRCDEVID1, Device Configuration Register 1

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

ETE

Register offset

0xFC4

Reset value

0x0

Bit descriptions

Figure B-117: ext_trcdevid1 bit assignments

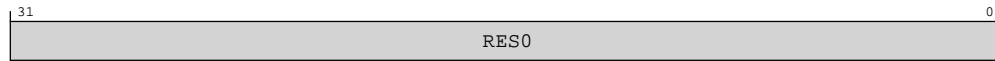


Table B-123: TRCDEVID1 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.6.23 TRCDEVID, Device Configuration Register

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFC8

Reset value

0x0

Bit descriptions

Figure B-118: ext_trcdevid bit assignments

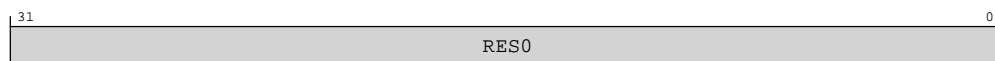


Table B-124: TRCDEVID bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.6.24 TRCDEVTYPE, Device Type Register

Provides discovery information for the component. If the part number field is not recognized, a debugger can report the information that is provided by TRCDEVTYPE about the component instead.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFCC

Reset value

See individual bit resets

Bit descriptions

Figure B-119: ext_trcdevtype bit assignments



Table B-125: TRCDEVTYPE bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | SUB | Component sub-type. 0b0001 When MAJOR == 0x3 (Trace source): Associated with a PE. This field reads as 0x1. | |
| [3:0] | MAJOR | Component major type. 0b0011 Trace source. Other values are defined by the CoreSight Architecture. This field reads as 0x3. | |

B.6.25 TRCPIDR4, Peripheral Identification Register 4

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD0

Reset value

See individual bit resets

Bit descriptions

Figure B-120: ext_trcpidr4 bit assignments



Table B-126: TRCPIDR4 bit descriptions

| Bits | Name | Description | Reset |
|--------|-------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | SIZE | The component uses a single 4KB block. 0b0000 | |
| [3:0] | DES_2 | Arm Limited. This is bits[3:0] of the JEP106 continuation code. 0b0100 | |

B.6.26 TRCPIDR5, Peripheral Identification Register 5

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

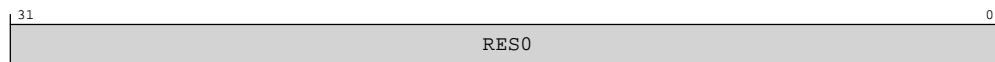
ETE

Register offset

0xFD4

Reset value

0x0

Bit descriptions**Figure B-121: ext_trcpidr5 bit assignments****Table B-127: TRCPIDR5 bit descriptions**

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.6.27 TRCPIDR6, Peripheral Identification Register 6

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

ETE

Register offset

0xFD8

Reset value

0x0

Bit descriptions

Figure B-122: ext_trcpidr6 bit assignments

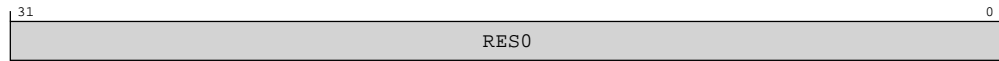


Table B-128: TRCPIDR6 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.6.28 TRCPIDR7, Peripheral Identification Register 7

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFDC

Reset value

0x0

Bit descriptions

Figure B-123: ext_trcpidr7 bit assignments

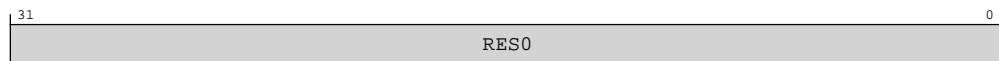


Table B-129: TRCPIDR7 bit descriptions

| Bits | Name | Description | Reset |
|--------|------|-------------|-------|
| [31:0] | RES0 | Reserved | 0x0 |

B.6.29 TRCPIDR0, Peripheral Identification Register 0

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE0

Reset value

See individual bit resets

Bit descriptions

Figure B-124: ext_trcpidr0 bit assignments



Table B-130: TRCPIDR0 bit descriptions

| Bits | Name | Description | Reset |
|--------|--------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PART_0 | Least significant byte of the trace unit part. 0b01001111 | |

B.6.30 TRCPIDR1, Peripheral Identification Register 1

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

ETE

Register offset

0xFE4

Reset value

See individual bit resets

Bit descriptions**Figure B-125: ext_trcpidr1 bit assignments****Table B-131: TRCPIDR1 bit descriptions**

| Bits | Name | Description | Reset |
|--------|--------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | DES_0 | Arm Limited. This is the least significant nibble of JEP106 ID code. 0b1011 | |
| [3:0] | PART_1 | Part number, most significant nibble. 0b1101 | |

B.6.31 TRCPIDR2, Peripheral Identification Register 2

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

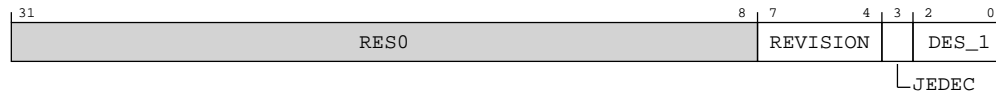
ETE

Register offset

0xFE8

Reset value

See individual bit resets

Bit descriptions**Figure B-126: ext_trcpidr2 bit assignments****Table B-132: TRCPIDR2 bit descriptions**

| Bits | Name | Description | Reset |
|--------|----------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | REVISION | rOp1 - Part major revision. 0b0001 | |
| [3] | JEDEC | JEDEC-assigned JEP106 implementer code is used. 0b1 RES1. Indicates a JEP106 identity code is used | |
| [2:0] | DES_1 | Arm Limited. Most significant nibble of JEP106 ID code. 0b011 | |

B.6.32 TRCPIDR3, Peripheral Identification Register 3

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

ETE

Register offset

0xFEC

Reset value

See individual bit resets

Bit descriptions

Figure B-127: ext_trcpidr3 bit assignments



Table B-133: TRCPIDR3 bit descriptions

| Bits | Name | Description | Reset |
|--------|--------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | REVAND | Part minor revision. 0b0000 | |
| [3:0] | CMOD | Not Customer modified. 0b0000 | |

B.6.33 TRCCIDR0, Component Identification Register 0

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF0

Reset value

See individual bit resets

Bit descriptions

Figure B-128: ext_trccidr0 bit assignments



Table B-134: TRCCIDR0 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|---|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_0 | Component identification preamble, segment 0. 0b00001101 Preamble byte 0 | |

B.6.34 TRCCIDR1, Component Identification Register 1

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF4

Reset value

See individual bit resets

Bit descriptions

Figure B-129: ext_trccidr1 bit assignments**Table B-135: TRCCIDR1 bit descriptions**

| Bits | Name | Description | Reset |
|--------|---------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:4] | CLASS | Component class. 0b1001 CoreSight peripheral. | |
| [3:0] | PRMBL_1 | Component identification preamble, segment 1. 0b0000 Preamble | |

B.6.35 TRCCIDR2, Component Identification Register 2

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF8

Reset value

See individual bit resets

Bit descriptions

Figure B-130: ext_trccidr2 bit assignments



Table B-136: TRCCIDR2 bit descriptions

| Bits | Name | Description | Reset |
|--------|---------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_2 | Component identification preamble, segment 2. 0b00000101 Preamble byte 2. | |

B.6.36 TRCCIDR3, Component Identification Register 3

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes**Width**

32

Component

ETE

Register offset

0xFFC

Reset value

See individual bit resets

Bit descriptions**Figure B-131: ext_trccidr3 bit assignments****Table B-137: TRCCIDR3 bit descriptions**

| Bits | Name | Description | Reset |
|--------|---------|--|-------|
| [31:8] | RES0 | Reserved | 0x0 |
| [7:0] | PRMBL_3 | Component identification preamble, segment 3. 0b10110001 Preamble byte 3. | |

Appendix C Document revisions

This appendix records the changes between released issues of this document.

C.1 Revisions

Changes between released issues of this book are summarized in tables.

The first table is for the first release. Then, each table compares the new issue of the book with the last released issue of the book. Release numbers match the revision history in [Release Information](#) on page 2.

Table C-1: Issue 0000-01

| Change | Location |
|-------------------------------------|----------|
| First early access release for r0p0 | - |

Table C-2: Differences between issue 0000-01 and issue 0001-02

| Change | Location |
|---|--|
| First early access release for r0p1 | Revision history |
| Updated powerup and powerdown sequence | 5.6 Neoverse V2 core powerup and powerdown sequence on page 47 |
| Updated write streaming mode | 8.5 Write streaming mode on page 66 |
| Added description on FEAT_ECBHB | 2.4 Supported standards and specifications on page 25 |
| Updated L1 data TLB format for Data Register 2 to 48 bit PA | 10.1.7 L1 data TLB returned data on page 77 |
| Updated L1 TRBE TLB in MMU components table | 6.1 Memory Management Unit components on page 51 |
| Added maximum limit to Transaction capabilities paragraph | 9.3 Transaction capabilities on page 69 |