Arm® C1-Scalable Matrix Extension 2

Revision r1p2

Technical Reference Manual

Non-Confidential

Copyright © 2023–2025 Arm Limited (or its affiliates).
All rights reserved.

Issue 06
107831_0102_06_en

Arm® C1-Scalable Matrix Extension 2 Technical Reference Manual

This document is Non-Confidential.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.

This document is protected by copyright and other intellectual property rights.
Arm only permits use of this document if you have reviewed and accepted Arm's Proprietary Notice found at the end of this document.

This document (107831_0102_06_en) was issued on 2025-09-10. There might be a later issue at https://developer.arm.com/documentation/107831

The product revision is r1p2.

See also: Proprietary Notice | Product and document information | Useful resources

Start reading

If you prefer, you can skip to the start of the content.

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.

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.

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.

Contents

1. The C1-SME2 unit 9
1.1 C1-SME2 unit features 11
1.1.1 Streaming execution priority 12
1.1.2 Arbitration 13
1.1.3 Fast context switch instructions 14
1.1.4 Accumulation array 14
1.2 C1-SME2 configuration options 14
1.3 C1-DSU dependent features 15
1.4 Supported standards and specifications 15
1.5 Test features 23
1.6 Design tasks 23
1.7 Product revisions 24

2. Technical overview 25
2.1 C1-SME2 unit components 25
2.2 Interfaces 28
2.3 Programmer's model 28

3. Clocks and resets 29

4. Power management 30
4.1 Voltage and power domains 30
4.2 Architectural clock gating modes 32
4.2.1 Low-power mode 32
4.2.2 Low-power mode behavior considerations 33
4.3 Power control 33
4.4 C1-SME2 unit power modes 34
4.4.1 On mode 36
4.4.2 Off mode 36
4.4.3 Emulated off mode 37
4.4.4 Debug recovery mode 37
4.4.5 Warm reset mode 38
4.5 Performance and power management 39

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

4.5.1 Maximum Power Mitigation Mechanism.....39

5. Memory management.....40
5.1 Translation Lookaside Buffer entry content.....40
5.2 Memory behavior and supported memory types.....40

6. L1 data memory system.....43
6.1 L1 data cache behavior.....43
6.2 Write streaming capability.....44
6.3 Data prefetching.....44

7. Direct access to internal memory.....46
7.1 L1 data cache encodings.....46

8. RAS Extension support.....48
8.1 Cache protection behavior.....49
8.2 Error containment.....49
8.3 Fault detection and reporting.....50
8.4 Error injection.....51
8.5 AArch64 RAS registers.....52
8.6 External RAS registers.....52

9. Utility bus.....54
9.1 Base addresses for system components.....54

10. System control.....56
10.1 Accessing the System registers.....56
10.2 Arbitration control.....57
10.3 Clock gate and power control.....59
10.4 AArch64 Generic System Control registers.....59

11. Debug.....61
11.1 Debug register interfaces.....62
11.1.1 C1-SME2 unit interfaces.....62
11.2 Debug memory map and debug signals.....63
11.3 ROM table.....63
11.4 CoreSight component identification.....64
11.5 External ROM table registers.....64

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

12. Performance Monitors Extension support ... 65
12.1 Common performance monitoring unit events ... 65
12.2 Implementation-defined performance monitoring unit events ... 73

13. Activity Monitors Extension support ... 88
13.1 Activity monitors access ... 88
13.2 Activity monitors counters ... 89
13.3 Activity monitors events ... 89
13.4 External AMU registers ... 90

14. Statistical Profiling Extension support ... 92

A. AArch64 registers ... 93
A.1 AArch64 Generic System Control registers summary ... 93
A.1.1 IMP_CMECFR_EL1, SME2 Features Register ... 93
A.1.2 IMP_CMESELR_EL1, SME2 Selection Register ... 96
A.1.3 IMP_CMEACTLR_EL1, SME2 Auxiliary Control Register ... 98
A.1.4 IMP_CMEACTLR2_EL1, SME2 Auxiliary Control Register 2 ... 100
A.1.5 IMP_CMECFG_EL1, SME2 Configuration Register ... 102
A.1.6 IMP_CMEPWR_EL1, SME2 Power Register ... 106
A.1.7 IMP_CMEREVIDR_EL1, SME2 unit ECO ID Register ... 108
A.1.8 IMP_CABACTLR_EL1, CAB Auxiliary Control Register ... 109
A.1.9 IMP_CABECTLR_EL1, CAB Control Register ... 112
A.1.10 IMP_CMERAMDATA0_EL3, Data0 for RAMINDEX ... 114
A.1.11 IMP_CMERAMDATA1_EL3, Data1 for RAMINDEX ... 116
A.1.12 IMP_CMERAMDATA2_EL3, Data2 for RAMINDEX ... 118
A.2 AArch64 Identification registers summary ... 120
A.2.1 SMIDR_EL1, Streaming Mode Identification Register ... 121
A.2.2 IMP_CMEMPMMCR_EL3, Global MPMM Configuration Register ... 123
A.3 AArch64 RAS registers summary ... 130
A.3.1 ERRIDR_EL1, Error Record ID Register ... 131
A.3.2 ERRSELR_EL1, Error Record Select Register ... 133
A.3.3 ERXFR_EL1, Selected Error Record Feature Register ... 135
A.3.4 ERXCTLR_EL1, Selected Error Record Control Register ... 137
A.3.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register ... 139
A.3.6 ERXADDR_EL1, Selected Error Record Address Register ... 142
A.3.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature Register ... 144

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

A.3.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control Register.... 146
A.3.9 ERXPFGCDN_EL1, Selected Pseudo-fault Generation Countdown Register.... 149
A.3.10 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0.... 152
A.3.11 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1.... 154
A.3.12 ERXMISC2_EL1, Selected Error Record Miscellaneous Register 2.... 156
A.3.13 ERXMISC3_EL1, Selected Error Record Miscellaneous Register 3.... 159
A.4 AArch64 Special-purpose registers summary.... 161
A.4.1 IMP_CMEPPMCR_EL3, Global PPM Configuration Register.... 162
A.4.2 IMP_CMEMPMMTUNE_EL3, MPMM Tuning Configuration Register.... 164
A.5 AArch64 System instructions summary.... 167
A.5.1 SYS_IMP_CMERAMINDEX, SYS_IMP_CMERAMINDEX system instruction.... 168

B. External registers.... 171
B.1 External AMU registers summary.... 171
B.1.1 AMEVCNTR00, Activity Monitors Event Counter Registers 0.... 172
B.1.2 AMEVCNTR01, Activity Monitors Event Counter Registers 0.... 173
B.1.3 AMEVCNTR02, Activity Monitors Event Counter Registers 0.... 175
B.1.4 AMEVCNTR03, Activity Monitors Event Counter Registers 0.... 176
B.1.5 AMEVCNTR10, Activity Monitors Event Counter Registers 1.... 177
B.1.6 AMEVCNTR11, Activity Monitors Event Counter Registers 1.... 179
B.1.7 AMEVCNTR12, Activity Monitors Event Counter Registers 1.... 180
B.1.8 AMEVCNTR13, Activity Monitors Event Counter Registers 1.... 181
B.1.9 AMEVTYPER00, Activity Monitors Event Type Registers 0.... 183
B.1.10 AMEVTYPER01, Activity Monitors Event Type Registers 0.... 184
B.1.11 AMEVTYPER02, Activity Monitors Event Type Registers 0.... 186
B.1.12 AMEVTYPER03, Activity Monitors Event Type Registers 0.... 187
B.1.13 AMEVTYPER10, Activity Monitors Event Type Registers 1.... 189
B.1.14 AMEVTYPER11, Activity Monitors Event Type Registers 1.... 190
B.1.15 AMEVTYPER12, Activity Monitors Event Type Registers 1.... 192
B.1.16 AMEVTYPER13, Activity Monitors Event Type Registers 1.... 193
B.1.17 AMCNTENSET0, Activity Monitors Count Enable Set Register 0.... 195
B.1.18 AMCNTENSET1, Activity Monitors Count Enable Set Register 1.... 197
B.1.19 AMCNTENCLR0, Activity Monitors Count Enable Clear Register 0.... 199
B.1.20 AMCNTENCLR1, Activity Monitors Count Enable Clear Register 1.... 201
B.1.21 AMCGCR, Activity Monitors Counter Group Configuration Register.... 204
B.1.22 AMCFGR, Activity Monitors Configuration Register.... 205

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.1.23 AMCR, Activity Monitors Control Register....207
B.1.24 AMIIDR, Activity Monitors Implementation Identification Register....208
B.1.25 AMDEVAFF0, Activity Monitors Device Affinity Register 0....210
B.1.26 AMDEVAFF1, Activity Monitors Device Affinity Register 1....211
B.1.27 AMDEVARCH, Activity Monitors Device Architecture Register....212
B.1.28 AMDEVTYPE, Activity Monitors Device Type Register....213
B.1.29 AMPIDR4, Activity Monitors Peripheral Identification Register 4....215
B.1.30 AMPIDR0, Activity Monitors Peripheral Identification Register 0....216
B.1.31 AMPIDR1, Activity Monitors Peripheral Identification Register 1....217
B.1.32 AMPIDR2, Activity Monitors Peripheral Identification Register 2....219
B.1.33 AMPIDR3, Activity Monitors Peripheral Identification Register 3....220
B.1.34 AMCIDR0, Activity Monitors Component Identification Register 0....221
B.1.35 AMCIDR1, Activity Monitors Component Identification Register 1....223
B.1.36 AMCIDR2, Activity Monitors Component Identification Register 2....224
B.1.37 AMCIDR3, Activity Monitors Component Identification Register 3....225
B.2 External MPMM registers summary....226
B.2.1 CMEPPMCR, Global PPM Configuration Register....227
B.2.2 CMEMPMMCR, Global MPMM Configuration Register....229
B.2.3 CMEMPMMTUNE, MPMM Tuning Configuration Register....236
B.3 External RAS registers summary....238
B.3.1 ERR0FR, Error Record <n> Feature Register....239
B.3.2 ERR0CTLR, Error Record <n> Control Register....243
B.3.3 ERR0STATUS, Error Record <n> Primary Status Register....246
B.3.4 ERR0ADDR, Error Record <n> Address Register....255
B.3.5 ERR0MISC0, Error Record <n> Miscellaneous Register 0....257
B.3.6 ERR0MISC1, Error Record <n> Miscellaneous Register 1....261
B.3.7 ERR0MISC2, Error Record <n> Miscellaneous Register 2....263
B.3.8 ERR0MISC3, Error Record <n> Miscellaneous Register 3....265
B.3.9 ERR0PFGF, Error Record <n> Pseudo-fault Generation Feature Register....267
B.3.10 ERR0PFGCTL, Error Record <n> Pseudo-fault Generation Control Register....270
B.3.11 ERR0PFGCDN, Error Record <n> Pseudo-fault Generation Countdown Register....273
B.3.12 ERRGSR, Error Group Status Register....275
B.3.13 ERRIIDR, Implementation Identification Register....277
B.3.14 ERRDEVAFF, Device Affinity Register....278
B.3.15 ERRDEVARCH, Device Architecture Register....280
B.3.16 ERRDEVID, Device Configuration Register....282

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.3.17 ERRPIDR4, Peripheral Identification Register 4....283
B.3.18 ERRPIDR0, Peripheral Identification Register 0....285
B.3.19 ERRPIDR1, Peripheral Identification Register 1....286
B.3.20 ERRPIDR2, Peripheral Identification Register 2....288
B.3.21 ERRPIDR3, Peripheral Identification Register 3....289
B.3.22 ERRCIDR0, Component Identification Register 0....291
B.3.23 ERRCIDR1, Component Identification Register 1....292
B.3.24 ERRCIDR2, Component Identification Register 2....293
B.3.25 ERRCIDR3, Component Identification Register 3....295
B.4 External ROM table registers summary....296
B.4.1 ROMENTRY0, Class 0x9 ROM Table Entries....297
B.4.2 AUTHSTATUS, Authentication Status Register....299
B.4.3 DEVARCH, Device Architecture Register....300
B.4.4 DEVID, Device Configuration Register....302
B.4.5 PIDR4, Peripheral Identification Register 4....303
B.4.6 PIDR0, Peripheral Identification Register 0....304
B.4.7 PIDR1, Peripheral Identification Register 1....305
B.4.8 PIDR2, Peripheral Identification Register 2....306
B.4.9 PIDR3, Peripheral Identification Register 3....308
B.4.10 CIDR0, Component Identification Register 0....309
B.4.11 CIDR1, Component Identification Register 1....310
B.4.12 CIDR2, Component Identification Register 2....311
B.4.13 CIDR3, Component Identification Register 3....312

Proprietary Notice....314

Product and document information....316
Product status....316
Revision history....316
Conventions....318

Useful resources....321

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

1. The C1-SME2 unit

The C1-SME2 unit is a shared unit that implements the Scalable Matrix Extension (SME) architecture, the Scalable Matrix Extension 2 (SME2) architecture, and the Arm®v9.3-A architecture. The Arm®v9.3-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.8-A.

SME and SME2 define architectural states capable of holding two-dimensional matrix tiles and a Streaming SVE (SSVE) mode for the cores.

Note

The C1-SME2 unit (previously known as the CME unit) is also referred to as the SME2 unit throughout this manual. However, some instances of the term CME (register bits and PMU events) remain in this manual.

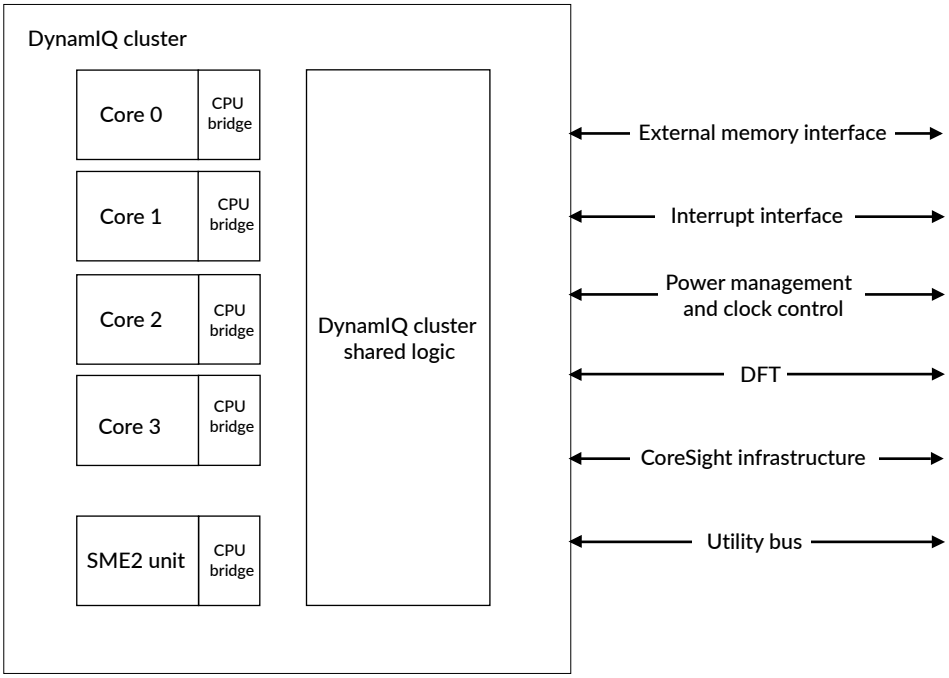
The C1-SME2 unit is implemented inside a C1-DSU cluster. It is a shared unit between all the cores and is accessible through the C1-DynamIQ™ Shared Unit (DSU), that behaves as a full interconnect with shared L3 support and full coherency between the cores and the C1-SME2 unit.

The C1-DSU cluster can support up to two C1-SME2 unit instances. If there are two C1-SME2 unit instances, the cores use a central CME Assignment Block (CAB) to request which C1-SME2 unit instance they should connect to.

The following figure shows an example configuration with one C1-SME2 unit and four cores in a DynamIQ™ cluster.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 1-1: Example C1-DSU cluster with one C1-SME2 unit



The C1-SME2 unit communicates with the cores using the AXI-Stream channels:

- CMEREQTX/CMEREQRX: streaming request channel
- CMETLBTX/CMETLBRX: Translation Lookaside Buffer (TLB) translation request channel

---

Note

- This manual applies to the C1-SME2 unit only. Read this manual together with the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual* for detailed information about the C1-DSU and with your core *Technical Reference Manual* (TRM) for detailed information about your core.
- For more information on the SVE and SVE2 extensions, see the *Arm® Architecture Reference Manual for A-profile architecture*.
- For more information on the SME and SME2 extensions, see the *Arm® Architecture Reference Manual Supplement, The Scalable Matrix Extension (SME), for Armv9-A*.
- This manual only lists registers that are relevant for the C1-SME2 unit. Read this manual together with your C1-DSU TRM, core TRM, and the *Arm® Architecture Reference Manual for A-profile architecture*.

Streaming SVE mode

The Streaming SVE (SSVE) mode supports execution of Scalable Vector Extension (SVE) instructions with a vector length that matches the tile width, called Streaming SVE Vector Length (SVL). The SVL can be different from the core SVE Vector Length (VL).

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When the core is in SSVE mode, SVE instructions execute inside the C1-SME2 unit with the SVL vector length, fixed to 512 bits. The SME and SME2 instructions and a subset of the Advanced SIMD and SVE instructions can be executed.

When the core is not in SSVE mode, SVE instructions execute inside the core with the VL vector length, fixed to 128 bits. The SME and SME2 instructions are Undefined.

1.1 C1-SME2 unit features

You can use the C1-SME2 unit in a DynamIQ™ configuration where your C1-DSU cluster includes one or two C1-SME2 instances and one or more cores.

Regardless of the cluster configuration, the C1-SME2 unit supports the following features:

C1-SME2 unit features

- Compatibility with the Arm®v9.3-A A64 instruction set
- AArch64 Execution state at all Exception levels from EL0 to EL3
- 40-bit Physical Address (PA) and 48-bit Virtual Address (VA)
- Implementation of the Reliability, Availability, and Serviceability (RAS) extension
- Implementation of a subset of the Scalable Vector Extension (SVE) and Scalable Vector Extension 2 (SVE2) instructions in Streaming SVE mode with a 512-bit vector length, as defined in the Scalable Matrix Extension (SME) and Scalable Matrix Extension 2 (SME2) architectures
- Implementation of the Scalable Matrix Extension (SME) and Scalable Matrix Extension 2 (SME2) with a 512-bit vector length
- Activity Monitoring Unit (AMU)

Cache features

- Private L1 data cache
- Optional error protection with Error Correcting Code (ECC) allowing Single Error Correction and Double Error Detection (SECDED) when a RAM instance can hold dirty data
- Support for Memory System Resource Partitioning and Monitoring (MPAM)

Debug features

- Arm®v8.8 debug architecture
- Performance Monitoring Unit (PMU)
- Statistical Profiling Extension (SPE)
- Support for the optional Embedded Logic Analyzer (ELA), ELA-600

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

The ELA-600 is licensed separately.

Related information

2. Technical overview on page 25

1.1.1 Streaming execution priority

Scalable Matrix Extension (SME) architecture introduces the streaming execution priority.

The SME architecture defines streaming execution priority through two architectural System registers:

- Streaming Mode Priority Register (SMPRI_EL1) configures the streaming execution priority for instructions executed on the C1-SME2 unit when the core is in Streaming SVE mode at any Exception Level.
- Streaming Mode Priority Mapping Register (SMPRIMAP_EL2) maps the value in SMPRI_EL1 to a streaming execution priority value for instructions executed at EL1and EL0 in the same Security states as EL2. The meaning of the priorities is **IMPLEMENTATION DEFINED**, with priority 0 being the lowest streaming execution priority and priority 15 being the highest streaming execution priority.

The C1-SME2 unit implements two types of priorities for the available 16 streaming execution priorities, Exclusive priority, and Fair-share priority.

Exclusive priorities

A core is guaranteed to get access to the C1-SME2 unit if that core uses higher Exclusive streaming execution priority relative to other cores that share the same C1-SME2 unit and use lower streaming execution priorities.

For the cores that share the same C1-SME2 unit, the cores that use lower streaming execution priorities than a core that uses Exclusive streaming execution priority are not guaranteed to forward progress, that is they might stall. If a core uses Exclusive streaming execution priority, the operating system needs to make sure to stop executing instructions on the C1-SME2 unit by clearing the PSTATE.SM bit to 0 (SMSTOP instruction) so that the other cores with lower streaming execution priorities do not stall.

A system may not give access to Exclusive streaming execution priorities to EL1 operating systems because the use of the Exclusive streaming execution priority might cause the cores with lower streaming execution priorities to stall. The system register SMPRIMAP_EL2 can be used for this purpose. For more information on the SMPRIMAP_EL2 register, see *Arm® Architecture Reference Manual for A-profile architecture*.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

For the cores that share the same C1-SME2 unit, if several cores share the same Exclusive streaming execution priority and no other core uses a higher Exclusive streaming execution priority, then:

- The arbitration between these cores is round-robin
- The cores get access to equal time slots

For more information on the arbitration behavior, see 1.1.2 Arbitration on page 13 and 10.2 Arbitration control on page 57.

Only a selection of the higher streaming execution priorities use Exclusive priority. This is controlled by FAIRSHARE_UPPER field of the A.1.5 IMP_CMECFG_EL1, SME2 Configuration Register on page 102.

Fair-share priorities

When several cores use Fair-share streaming execution priorities, and no core has selected an Exclusive streaming execution priority, forward progress is guaranteed even when different Fair-share streaming execution priorities are selected.

Arbitration between cores using Fair-share streaming execution priorities is round-robin.

Once a core is selected for arbitration, the priority acts on the maximum allocated time slot. Higher Fair-share streaming execution priority gets more time slot than lower Fair-share streaming execution priority.

1.1.2 Arbitration

When several cores try to access the same C1-SME2 unit instance, the C1-SME2 unit arbitrates between the requesting cores. The arbitration depends on the streaming execution priorities.

A.1.5 IMP_CMECFG_EL1, SME2 Configuration Register on page 102 controls the switching between the cores. For more information on how different fields impact the arbitration behavior, see 10.2 Arbitration control on page 57 .

Each core that can be arbitrated runs for a maximum number of cycles, independently of SMSTOP and SMSTART instructions. However, the C1-SME2 unit supports idle detection, that is if a core does not send instructions for some time, the C1-SME2 unit asks the arbitrated core to disconnect and provides access to other cores.

Note

When a core is halted in Debug State and rarely sends instructions to the C1-SME2 unit, the other cores are able to progress even if the halted core has higher streaming execution priority.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

1.1.3 Fast context switch instructions

Fast context switch instructions do not depend on the streaming execution priority (SMPRI_EL1 and SMPRIMAP_EL2) and arbitration to complete.

Fast context switch instructions are used by the Operating System when a process executing instructions in the Streaming mode is interrupted. This guarantees that the interrupt handler is not stalled because of the arbitration. It is assumed that only fast context switch instructions are used to save or restore the architectural context of a software process.

Fast context switch instructions executed from one core can be interleaved with instructions executed from another core. No data is shared between the two cores, including:

- Any architectural state (Z, ZA, or ZT0 registers)
- Merging of data which is not visible at the point of coherence yet, typically in the merge buffer

When a core is stopped to give priority to another core, its architectural context (Z, ZA, ZT0, P) is saved to Context storage. The architectural state needs to be correctly maintained and restored.

1.1.4 Accumulation array

Additional architectural state storage is provided by an accumulation array of registers called ZA, organized in tiles, and a ZT0 register that supports a lookup table feature.

The ZA array permits outer product and accumulation of two Scalable Vector Extension (SVE) vectors into a matrix with a single instruction. Multi-vector multiply-accumulate operations are also supported. The accumulation matrix sums the result of one iteration to the previous result. For more information, see *Arm® Architecture Reference Manual for A-profile architecture*.

1.2 C1-SME2 configuration options

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

Configuration is on a per C1-SME2 unit basis. However if you have two C1-SME2 unit instances, they should have the exact same configuration that is, each configuration option should have the same value for both C1-SME2 unit instances.

You can configure your C1-SME2 unit implementation using the following options:

Cache protection

You can configure your implementation with or without cache protection.

CoreSight™ Embedded Logic Analyzer (ELA)

You can include support for integrating the ELA-600 as a separate licensable product. If the ELA-600 is included, you can also configure the ATB FIFO depth.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Matrix execute throughput

You can configure your C1-SME2 Matrix execute unit as half-matmul or full-matmul. For more information, see 2.1 C1-SME2 unit components on page 25.

Number of CHI interfaces

You can configure the number of CHI interfaces between the C1-SME2 unit and the C1-DynamIQ™ Shared Unit (DSU).

Number of cores

You can configure the number of cores that the C1-SME2 unit supports.

For detailed configuration options and guidelines, see *RTL configuration process* in the *Arm® C1-Scalable Matrix Extension 2 Configuration and Integration Manual*.

1.3 C1-DSU dependent features

Some C1-DynamIQ™ Shared Unit (DSU) features and behaviors depend on whether the C1-SME2 unit supports a particular feature.

The following table describes which C1-DSU dependent features are supported in your C1-SME2 unit.

Table 1-1: C1-SME2 features that have a dependency on the C1-DSU

Feature	Supported in the C1-SME2 unit	Dependency on the C1-DSU
Direct connect	No	-
Maximum Power Mitigation Mechanism (MPMM)	Yes	Affects the external signals of the C1-DSU. For more information on MPMM, see 4.5 Performance and power management on page 38
Statistical Profiling Extension (SPE) architecture	Yes	Affects the external signals of the C1-DSU.
Physical Address (PA) width	40-bit	Affects the CHI requester and AXI manager port bus widths. For more details, see the following chapters of the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*: • *CHI requester interface* • *AXI manager interface*
Number of cores in the cluster	1-14	Impacts the size of the Context storage RAMs.

1.4 Supported standards and specifications

The C1-SME2 unit implements the Arm®v9.3-A architecture. The Arm®v9.3-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.8-A. The C1-SME2 unit also implements specific Arm®v8-A and Arm®v9-A architecture extensions and supports

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

interconnect architectures. Debug and trace architectures are also supported through the core that the C1-SME2 unit is connected to.

The C1-SME2 unit supports AArch64 only at all Exception levels, EL0 to EL3.

The following tables show the features that the C1-SME2 unit implements for each of the Arm®v8-A and Arm®v9-A architecture versions.

Note

For more information on the features listed in the following tables, see the *Arm® Architecture Reference Manual for A-profile architecture*.

Table 1-2: Arm®v8.0-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_MixedEndEL0	Yes	Mixed-endian support at EL0
FEAT_MixedEnd	Yes	Mixed-endian support
FEAT_BigEndEL0	Yes	Big-endian support at EL0
FEAT_BigEnd	Yes	Big-endian support
FEAT_LittleEndEL0	Yes	Little-endian support at EL0
FEAT_LittleEnd	Yes	Little-endian support
FEAT_FP	Yes	Floating-point Extension
FEAT_AdvSIMD	Because FEAT_SME_FA64 is not implemented, only a subset of these instructions is supported in Streaming SVE mode as defined in the Scalable Matrix Extension (SME) and Scalable Matrix Extension 2 (SME2) architectures.	Advanced SIMD Extension For more information and register descriptions, see the *Arm® Architecture Reference Manual for A-profile architecture*.
FEAT_PMUv3	Yes	Performance Monitoring Unit (PMU) extension version 3
FEAT_PMUv3_EXT	Yes	External interface to the PMU
FEAT_PMUv3_EXT32	Yes	32-bit external interface to the PMU
FEAT_TRC_EXT	Yes	Trace external registers
FEAT_SB	Yes	Speculation barrier
FEAT_SSBS	Yes	Speculative Store Bypass Safe Instruction
FEAT_SSBS2	Yes	Speculative Store Bypass Safe Instruction version 2
FEAT_CSV2	Yes	Cache Speculation Variant 2
FEAT_CSV2_1p1	No	Cache Speculation Variant 2 version 1.1
FEAT_CSV2_1p2	No	Cache Speculation Variant 2 version 1.2
FEAT_CSV2_2	Yes	Cache Speculation Variant 2 version 2
FEAT_CSV2_3	Yes	Cache Speculation Variant 2 version 3
FEAT_CSV3	Yes	Cache Speculation Variant 3
FEAT_SPECRES	Yes	Speculation restriction instructions

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table 1-3: Arm®v8.1-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_LSE	Yes	Large System Extensions
FEAT_RDM	Yes	Rounding double multiply accumulate
FEAT_PAN	Yes	Privileged access-never
FEAT_LOR	Yes	Limited ordering regions
FEAT_HAFDBS	Yes	Hardware updates to access flag and dirty state in translation tables
FEAT_VMID16	Yes	16-bit VMID
FEAT_PMUv3p1	Yes	PMU extensions version 3.1
FEAT_PAN3	Yes	Support for SCTLR_ELx.EPAN

Table 1-4: Arm®v8.2-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_UAO	Yes	Unprivileged Access Override control
FEAT_DPB	Yes	DC CVAP instruction
FEAT_Debugv8p2	Yes	Arm®v8.2-A Debug
FEAT_IESB	Yes	Implicit Error synchronization event
FEAT_FP16	Yes	Half-precision floating-point data processing
FEAT_LVA	No	Large VA support
FEAT_LPA	No	Large PA and IPA support
FEAT_RAS	Yes	Reliability, Availability, and Serviceability (RAS) Extension version 1.1
FEAT_RASSA	Yes	RAS System Architecture
FEAT_RASSAv1	Yes	RAS version 1 System Architecture
FEAT_SpecSEI	No	SError interrupt exceptions from speculative reads of memory
FEAT_SPE	Yes	Statistical Profiling Extension (SPE) For more information, see 14. Statistical Profiling Extension support on page 92.
FEAT_SPE_LDS	Yes	Statistical Profiling data source packet generation
FEAT_SVE	Yes Because FEAT_SME_FA64 is not implemented, only a subset of these instructions is supported in Streaming SVE mode as defined in the SME and SME2 architectures.	Scalable Vector Extension (SVE)
FEAT_DotProd	Yes	Advanced SIMD Int8 dot product instructions
FEAT_FHM	Yes	Half-precision floating-point FMLAL instructions
FEAT_BF16	Yes	AArch64 BFloat16 instructions
FEAT_AA32BF16	No	AArch32 BFloat16 instructions
FEAT_I8MM	Yes	AArch64 Int8 matrix multiplication instructions

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Feature	Implemented	Description
FEAT_MPAM	Yes	Memory Partitioning and Monitoring (MPAM) For more information on the Memory System Resource Partitioning and Monitoring (MPAM) Extension, see the *Arm® Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A*.
FEAT_MPAMv1p0	Yes	Memory Partitioning and Monitoring Extension

Table 1-5: Arm®v8.3-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_JSCVT	Yes	JavaScript FJCVTS conversion instruction
FEAT_FCMA	Yes	Floating-point FCMLA and FCADD instructions
FEAT_SPEv1p1	Yes	Statistical Profiling Extensions version 1.1 For more information, see 14. Statistical Profiling Extension support on page 92.
FEAT_BBML3	No	Translation table break-before-make level 3

Table 1-6: Arm®v8.4-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_SEL2	Yes	Secure EL2
FEAT_S2FWB	Yes	Stage 2 forced write-back
FEAT_DIT	Yes	Data Independent Timing instructions
FEAT_LSE2	Yes	Large System Extensions version 2
FEAT_LRCPC2	Yes	Load-acquire RCpc instructions version 2
FEAT_TLBIOS	Yes	TLB invalidate outer-shared instructions
FEAT_TLBIRANGE	Yes	TLB range invalidate range instructions
FEAT_BBML1	Yes	Translation table break before make levels
FEAT_BBML2	Yes	Translation table break-before-make levels
FEAT_RASv1p1	Yes	Reliability, Availability, and Serviceability (RAS) Extension version 1.1
FEAT_RASSAv1p1	Yes	RAS version v1.1 System Architecture
FEAT_DoubleFault	Yes	Double Fault Extension
FEAT_Debugv8p4	Yes	Debug relaxations and extensions version 8.4
FEAT_PMUv3p4	Yes	PMU extension version 3.4
FEAT_TTST	Yes	Small translation tables
FEAT_AMUv1	Yes	Activity Monitors Extension version 1
FEAT_AMU_EXT	Yes	External Activity Monitors
FEAT_AMU_EXT32	Yes	32-bit External Activity Monitors extension
FEAT_AMU_EXTACR	No	Activity Monitors External Control Register

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table 1-7: Arm®v8.5-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_FlagM2	Yes	Condition flag manipulation version 2
FEAT_FRINTTS	Yes	FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions
FEAT_E0PD	Yes	Preventing EL0 access to halves of address maps
FEAT_MTE	Yes	Instruction-only Memory Tagging Extension The C1-SME2 core always implements the Memory Tagging Extension (MTE) and therefore is compliant with the CHI.E protocol. For information on CHI.E commands inferred by MTE, see the *CHI requester interface* chapter in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.
FEAT_MTE2	Yes, configurable	Memory Tagging Extension version 2
FEAT_MTE3	Yes, configurable	MTE Asymmetric Fault Handling
FEAT_MTE_ASYM_FAULT	Yes, configurable	Memory tagging asymmetric faults
FEAT_MTE_ASYNC	Yes, configurable	Memory Tagging asynchronous faulting
FEAT_PMUv3p5	Yes	PMU Extension version 3.5

Table 1-8: Arm®v8.6-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_AMUv1p1	No	Activity Monitors Extension version 1.1
FEAT_MPAMv0p1	No	Memory Partitioning and Monitoring version0.1
FEAT_MPAMv1p1	Yes	Memory Partitioning and Monitoring version1.1
FEAT_MTPMU	No	Multi-threaded PMU Extensions

Table 1-9: Arm®v8.7-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_AFP	Yes	Alternate floating-point behavior
FEAT_LPA2	No	Larger physical address for 4KB and 16KB translation granules
FEAT_PMUv3p7	Yes	Arm®v8.7-A PMU Extensions See 12.1 Common performance monitoring unit events on page 65.
FEAT_RPRES	No	Increased precision of Reciprocal Estimate and Reciprocal Square Root Estimate
FEAT_SPEv1p2	Yes	Arm®v8.7-A SPE See 14. Statistical Profiling Extension support on page 92.
FEAT_SPE_FnE	Yes	Statistical Profiling inverse event filter

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table 1-10: Arm®v8.8-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_Debugv8p8	Yes	Debug v8.8
FEAT_PMUv3p8	Yes	Arm®v8.8-A PMU Extensions
FEAT_SPEv1p3	Yes	Arm®v8.8-A Statistical Profiling Extensions
FEAT_S1PIE	No	Stage 1 permission indirections
FEAT_S2PIE	No	Stage 2 permission indirections
FEAT_S1POE	No	Stage 1 permission overlays
FEAT_S2POE	No	Permission model enhancements

Table 1-11: Arm®v8.9-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_ADERR	No	Asynchronous Device error exceptions
FEAT_AIE	No	Memory Attribute Index Enhancement
FEAT_AMU_EXT64	No	64-bit External Activity Monitors extension
FEAT_ANERR	No	Asynchronous Normal error exception
FEAT_ATS1A	No	Permission model enhancements
FEAT_CSSC	No	Common Short Sequence Compression instructions
FEAT_Debugv8p9	No	Debug v8.9
FEAT_DoubleFault2	No	Enhancements to the Double Fault Extension
FEAT_FGT2	No	Fine-grained traps 2
FEAT_MTE_CANONICAL_TAGS	No	Canonical Tag checking for Untagged memory
FEAT_MTE_NO_ADDRESS_TAGS	No	Memory tagging with Address tagging disabled
FEAT_MTE_STORE_ONLY	No	Store-only Tag Checking
FEAT_MTE_TAGGED_FAR	No	FAR_ELx on a Tag Check Fault
FEAT_MTE4	No	Enhanced Memory Tagging Extension
FEAT_PCSRv8p9	No	Armv8.9 PC Sample-based Profiling Extension
FEAT_PFAR	No	Physical Fault Address Registers
FEAT_PMUv3_EDGE	No	PMU event edge detection
FEAT_PMUv3_EXT64	No	64-bit external interface to the Performance Monitors
FEAT_PMUv3_ICNTR	No	Fixed-function instruction counter
FEAT_PMUv3_SS	No	PMU Snapshot Extension
FEAT_PMUv3p9	No	Armv8.9 PMU extensions
FEAT_PRFMSLC	No	SLC target support for PRFM instructions
FEAT_RASSAv2	No	RAS version 2 System Architecture
FEAT_RASv2	No	RAS version 2
FEAT_RPRFM	Yes	Support for Range Prefetch Memory instruction
FEAT_SPE_CRR	No	Call Return Branch Records
FEAT_SPE_FDS	No	Data Source Filtering
FEAT_SPEv1p4	No	Statistical Profiling Extension version 1.4

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Feature	Implemented	Description
FEAT_SPMU	No	System Performance Monitors Extension
FEAT_THE	No	Translation Hardening Extension
FEAT_RASSA_PARENT_GRP	No	System RAS agents
FEAT_RASSA_16KB_GRP	No	Large error record groups
FEAT_RASSA_64KB_GRP	No	Large error record groups
FEAT_RASSA_SRV	No	Error record reset
FEAT_RASSA_ACR	No	Security model for error records, and FEAT_RME
FEAT_RASSA_PFG_GRP	No	Security model for error records, and FEAT_RME
FEAT_RASSA_IRQCR_SIMPE	No	Simplified interrupt control registers
FEAT_RASSA_DFI	No	Reducing error interrupt reporting
FEAT_RASSA_RV	No	Error record reset
FEAT_RASSA_CED	No	Disable interrupts from error counters
FEAT_RASSA_ERT	No	Other error record changes

The following tables show the features that the C1-SME2 unit implements for each Arm®v9-A architecture version.

Table 1-12: Arm®v9.0-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_SVE2	Yes Because FEAT_SME_FA64 is not implemented, only a subset of these instructions is supported in Streaming SVE mode as defined in the SME and SME2 architectures.	Scalable Vector Extension (SVE) version 2
FEAT_TME	No	Transactional Memory Extension (TME)
FEAT_IDTE3	No	Trapping ID register accesses to EL3
FEAT_UINJ	No	Injection of Undefined Instruction exceptions
FEAT_CMH	No	Contention Management Hints
FEAT_EAESR	No	Enable Abort Exception System Registers
FEAT_NV3	No	Embedded Trace Extension

Table 1-13: Arm®v9.2-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_BRBE	No	Branch Record Buffer Extensions (BRBE)
FEAT_RME	No	Realm Management Extension (RME)
FEAT_ETEv1p2	No	Embedded Trace Extension (ETE) version 1.2
FEAT_SME	Yes	Scalable Matrix Extension (SME)
FEAT_SME_FA64	No	Full A64 support in Streaming mode
FEAT_SME_F64F64	No	Double-precision floating-point outer product instructions
FEAT_SME_I16I64	No	16-bit to 64-bit integer widening outer product instructions

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Feature	Implemented	Description
FEAT_EBF16	No	Enhanced BFloat16
FEAT_FAMINMAX	No	Floating-point (FP) maximum and minimum absolute value instructions
FEAT_FP8	No	FP8 convert instructions
FEAT_FP8DOT2	No	FP8 2-way dot product to half-precision instructions
FEAT_FP8DOT4	No	FP8 4-way dot product to single-precision instructions
FEAT_FP8FMA	No	FP8 multiply-accumulate to half-precision and single-precision instructions
FEAT_FPMR	No	FP Mode Register
FEAT_LUT	No	Lookup table instructions with 2-bit and 4-bit indices
FEAT_SME_F8F16	No	SME2 ZA-targeting FP8 multiply-accumulate, dot product, and outer product to half-precision instructions
FEAT_SME_F8F32	No	SME2 ZA-targeting FP8 multiply-accumulate, dot product, and outer product to single-precision instructions
FEAT_SME_LUTv2	No	Lookup table instructions with 4-bit indices and 8-bit elements
FEAT_SSVE_FP8DOT2	No	SVE2 FP8 2-way dot product to half-precision instructions in Streaming SVE mode
FEAT_SSVE_FP8DOT4	No	SVE2 FP8 4-way dot product to single-precision instructions in Streaming SVE mode
FEAT_SSVE_FP8FMA	No	SVE2 FP8 multiply-accumulate to half-precision and single-precision instructions in Streaming SVE mode
FEAT_SVE_BFSCALE	No	BFloat16 Floating-Point Adjust Exponent
FEAT_SVE_F16F32MM	No	SVE Half-Precision to Single-Precision Matrix Multiplication
FEAT_F8F32MM	No	FP8 to Single-Precision Matrix Multiplication
FEAT_F8F16MM	No	FP8 to Half-Precision Matrix Multiplication
FEAT_LS64WB	No	LS64 for Write-back cacheable memory
FEAT_SPE_SME	No	Statisical Profiling extensions for SME
FEAT_LORRL	No	Limited Order Regions in Realm PA space

Table 1-14: Arm®v9.3-A features implemented in the C1-SME2 unit

Feature	Implemented	Description
FEAT_BRBEv1p1	No	Arm®v9.3-A updates to BRBE
FEAT_SME2	Yes	SME version 2
FEAT_MEC	No	Memory Encryption Contexts
FEAT_MPAM_PE_BW_CTRL	No	MPAM PE-side Bandwidth Controls
FEAT_LSFE	No	Large System Float Extension
FEAT_LSCP	No	Load and Store Consistency Pair instructions
FEAT_TLBID	No	TLBI Domains

Related information

2.1 C1-SME2 unit components on page 25

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

1.5 Test features

The C1-SME2 unit provides test signals that enable the use of both Automatic Test Pattern Generation (ATPG) and Memory Built-In Self Test (MBIST) to test the unit logic and memory arrays.

The C1-SME2 unit includes an ATPG test interface that provides signals to control the Design for Test (DFT) features of the unit. To prevent problems with DFT implementation, you must carefully consider how you use these signals.

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

For the list of test signals and information on their usage, see the *Design for Test integration guidelines* chapter in the *Arm® C1-Scalable Matrix Extension 2 Configuration and Integration Manual*.

For the list of external scan control signals, see the *Design for Test integration guidelines* chapter in the *Arm® C1-DynamIQ™ Shared Unit Configuration and Integration Manual*.

Note

The *Arm® C1-Scalable Matrix Extension 2 Configuration and Integration Manual* and *Arm® C1-DynamIQ™ Shared Unit Configuration and Integration Manual* are confidential documents that are available with the appropriate product licenses.

1.6 Design tasks

The C1-SME2 unit is delivered as a synthesizable RTL description in SystemVerilog. Before you can use the C1-SME2 unit, you must implement, integrate, and program it.

A different party can perform each of the following tasks:

Implementation

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

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

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

Integration

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

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

The integrator configures some features of the C1-SME2 unit 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 unit and tests the application software.

The programmer configures the C1-SME2 unit by programming values into registers. The programmed values affect the behavior of the C1-SME2 unit.

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

For more information, see the following:

- *RTL configuration process* in the *Arm® C1-Scalable Matrix Extension 2 Configuration and Integration Manual* for implementation options.
- *RTL configuration process* in the *Arm® C1-DynamIQ™ Shared Unit Configuration and Integration Manual* for implementation options.
- *Functional integration* in the *Arm® C1-DynamIQ™ Shared Unit Configuration and Integration Manual* for signal descriptions.

1.7 Product revisions

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

Table 1-15: Product revisions

Revision	Notes
r0p0	First limited access release
r1p0	First early access release
r1p1	First early access release
r1p2	First early access release

Changes in functionality that have an impact on the documentation also appear in Revision history on page 316.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

2. Technical overview

The components in the C1-SME2 unit are designed to perform large vector and matrix processing in an energy-efficient way.

The main blocks include:

- Instruction decode
- Instruction rename
- Instruction issue
- Execution pipeline
- L1 data memory system
- Performance Monitoring Unit (PMU)
- Activity Monitoring Unit (AMU)

The C1-SME2 unit interfaces with the C1-DynamIQ™ Shared Unit (DSU) through the SME2 unit bridge.

The C1-SME2 unit implements the Scalable Matrix Extension (SME) and Scalable Matrix Extension 2 (SME2) architectures, and the Arm®v9.3-A architecture. The Arm®v9.3-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.8-A.

The programmer's model and the architecture features implemented are compliant with the standards in 1.4 Supported standards and specifications on page 15.

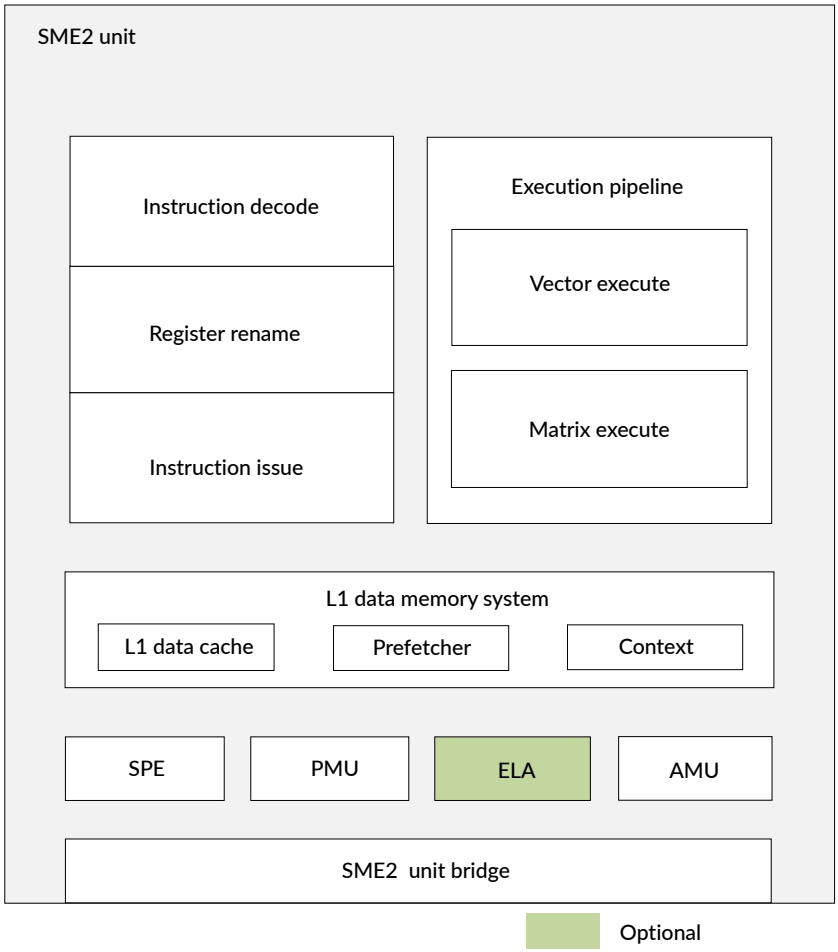
2.1 C1-SME2 unit components

The C1-SME2 unit includes components designed to perform large vector and matrix processing in an energy-efficient way. The C1-SME2 unit includes a SME2 unit bridge that connects the unit to the C1-DynamIQ™ Shared Unit (DSU). The C1-DSU connects the unit to an external memory system and the rest of the System on Chip (SoC).

The following figure shows the C1-SME2 unit components.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 2-1: C1-SME2 unit components



Instruction decode

The instruction decode unit decodes instructions from AArch64 into an internal format, which it then passes to the execution pipeline.

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.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Vector execute

The Vector execute unit is part of the execution pipeline and executes scalar and vector instructions operating on scalar registers, SVE registers (Z0-Z31), and the SME2 ZT0 register.

Matrix execute

The Matrix execute unit is part of the execution pipeline and executes vector instructions operating on ZA registers. It supports two configurations:

- Half-matmul, able to perform one outer product operation every other cycle
- Full-matmul, able to perform one outer product operation per cycle

L1 data memory system

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

The L1 data memory system includes:

- A 64KB, 4-way set associative L1 data cache with 64-byte cache lines
- A Prefetcher with 128 training streams
- A Context storage, to store all Z0-Z31, P0-P15, ZA, and ZT0 registers for each of the cores

Statistical Profiling Extension

The C1-SME2 unit implements the Statistical Profiling Extension (SPE) to the Arm®v8.8-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, 20, or 31 performance monitors within the cores that can be configured to gather statistics on the operation of each C1-SME2 unit and the memory system. The PMU can count events inside the C1-SME2 unit, but the actual counters are in the core and accessible by software. The PMU information can be used for debug and code profiling.

Activity Monitoring Unit

The C1-SME2 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.

SME2 unit bridge

In a cluster, there is one SME2 unit bridge between each C1-SME2 unit and the C1-DSU.

The SME2 unit bridge controls buffering and synchronization between the C1-SME2 unit and the C1-DSU.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

The SME2 unit bridge is asynchronous to allow different frequency, power, and area implementation points for each C1-SME2 unit. You can configure the SME2 unit bridge to run synchronously without affecting the other interfaces.

Related information

5. Memory management on page 40
6. L1 data memory system on page 43
12. Performance Monitors Extension support on page 65
13. Activity Monitors Extension support on page 88
14. Statistical Profiling Extension support on page 92

2.2 Interfaces

The C1-DynamIQ™ Shared Unit (DSU) manages all C1-SME2 external interfaces to the core.

For more information on the C1-SME2 external interfaces to the core, see the *Technical overview* chapter in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*

2.3 Programmer's model

The Arm®v9.3-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.8-A. The C1-SME2 supports the AArch64 Execution state at all Exception levels, EL0 to EL3.

The C1-SME2 implements the Scalable Matrix Extension (SME) and Scalable Matrix Extension 2 (SME2) architectures, and the Arm®v9.3-A architecture.

For more information about the programmer's model, SME, and SME2 see *Arm® Architecture Reference Manual for A-profile architecture*.

Related information

1.4 Supported standards and specifications on page 15

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3. Clocks and resets

To provide dynamic power savings, the C1-SME2 unit supports hierarchical clock gating and Warm and Cold resets.

Each C1-SME2 unit has a single clock domain and receives a single clock input. This clock input is gated by an architectural clock gate in the SME2 unit bridge.

The C1-SME2 unit also 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 C1-DynamIQ™ Shared Unit (DSU) side of the SME2 unit bridge provides two reset signals:

- A Warm reset for all registers in the C1-SME2 unit except for the AXI-Stream interface, Activity Monitoring Unit (AMU) logic, and a subset of Reliability, Availability, and Serviceability (RAS) logic
- A Cold reset for the logic in the C1-SME2 unit, including the AXI-Stream interface, AMU logic, and a subset of RAS logic.

For a complete description of the clock gating and reset scheme of the unit, see the following chapters in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*:

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

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

4. Power management

The C1-SME2 unit provides mechanisms to control both dynamic and static power dissipation.

The following features reduce the dynamic power dissipation in the C1-SME2 unit:

- Hierarchical clock gating
- Per C1-SME2 Dynamic Voltage and Frequency Scaling (DVFS)

Powerdown reduces the static power dissipation. When the Power Policy Unit (PPU) is programmed in Dynamic mode, powerdown is fully managed by the C1-SME2 unit.

Note

The C1-DynamIQ™ Shared Unit (DSU) Power Policy Units (PPUs) control power mode transitions, including powerdown, for the C1-SME2 unit. Your software can program a PPU to set a PPU mode in static or dynamic mode. If the PPU is in dynamic mode, power mode transitions, including powerdown, are requested automatically.

The C1-SME2 unit does not support debug over powerdown. However, debug is still possible through the core debug over powerdown while the C1-SME2 unit is powered down.

When there are two C1-SME2 units in the cluster, each unit is independent concerning power and clocking.

4.1 Voltage and power domains

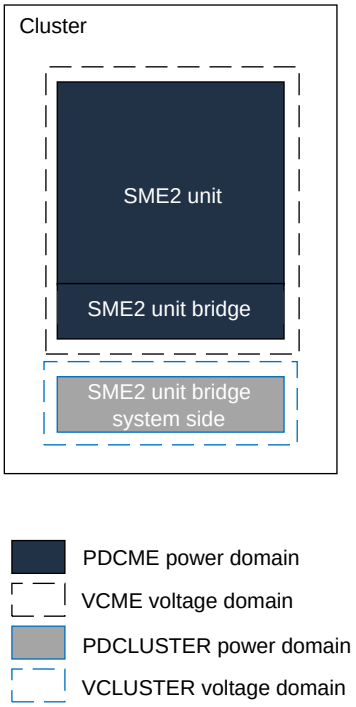
The C1-DynamIQ™ Shared Unit (DSU) Power Policy Units (PPUs) control power management for the C1-SME2 unit. The C1-SME2 unit supports one power domain, PDCME, and one system power domain, PDCLUSTER. Similarly, it supports one unit voltage domain, VCME, and one cluster system voltage domain, VCLUSTER. The power domains and voltage domains have the same boundaries.

The PDCME power domain contains all C1-SME2 logic and part of the unit asynchronous bridge that belongs to the VCME domain. The PDCLUSTER power domain contains the part of the SME2 unit bridge that belongs to the VCLUSTER domain.

The following figure shows the C1-SME2 power domain and voltage domain. It also shows the cluster power domain and voltage domain that cover the system side of the SME2 unit bridge.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 4-1: C1-SME2 power domains and voltage domains



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

- The C1-SME2 unit is configured to support Dynamic Voltage and Frequency Scaling (DVFS) with the same control as the C1-DSU
- The C1-SME2 unit is configured to run synchronously with the C1-DSU sharing the same clock.
- The C1-SME2 unit is not required to support DVFS

Note

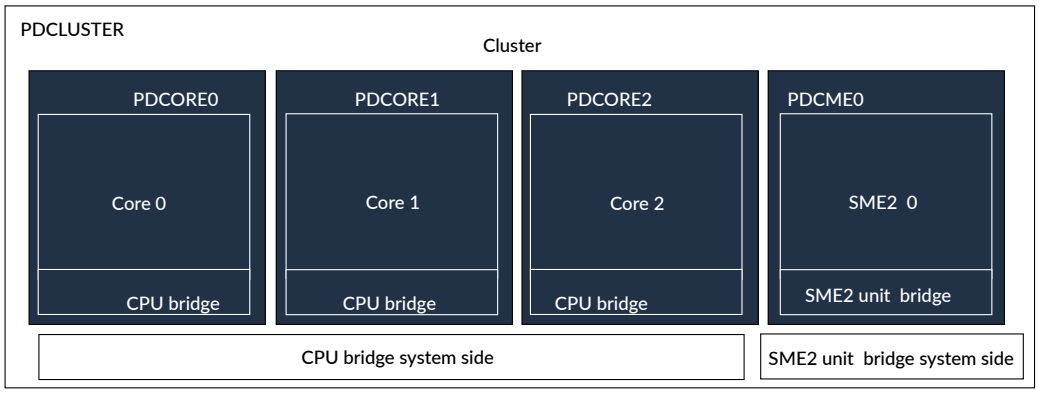
Even if the C1-SME2 unit is configured to use the same voltage rail as another unit, the C1-SME2 unit operates in a separate power domain enabling the powerdown of the unit to reduce static power dissipation.

In a cluster with two C1-SME2 units, there is one PDCME<n> power domain per unit, where n is the unit instance number. If a unit is not present, then the corresponding power domain is not present.

The following figure shows an example of the power domains in a cluster with three cores and a C1-SME2 unit.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 4-2: Power domains in a cluster with three cores and a C1-SME2 unit



Clamping cells between power domains are inferred through power intent files (UPF) rather than instantiated in the RTL. For more information, see *Power management* in the *Arm® C1-Scalable Matrix Extension 2 Configuration and Integration Manual* for more information.

For more information on the C1-DSU cluster power domains and voltage domains, see *Power management* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

4.2 Architectural clock gating modes

When the C1-SME2 unit enters into Standby mode, the transition is managed by a state machine.

Standby mode puts the C1-SME2 unit in a low-power mode by disabling most of the C1-SME2 unit clocks, while keeping the C1-SME2 unit powered up.

4.2.1 Low-power mode

Low-power mode behavior is handled in hardware by a state machine.

If the C1-SME2 unit has no internal activity, a timer is triggered with a programmable timeout that is controlled with the A.1.6 IMP_CMEPWR_EL1, SME2 Power Register on page 106. The unit is considered as having no internal activity when all instructions are completed and none of the cores send new instructions. When the timeout is reached, the C1-SME2 unit goes into Standby mode. While in Standby mode, the state machine tells the C1-DynamIQ™ Shared Unit (DSU) that it can stop the clocks. The state that the C1-SME2 unit requests depends on the powerdown conditions:

- If powerdown is permitted while the C1-SME2 unit is in Standby mode, the TIMEOUT_TO_OFF timeout counter starts counting. If the timeout period is completed without activity, the C1-SME2 unit requests a complete powerdown. For more information, see 4.3 Power control on page 33

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

- If powerdown is not permitted, the C1-SME2 unit stays in Standby mode.

When the C1-SME2 unit is in Standby mode, the input clock is gated externally to the C1-SME2 unit at the SME2 unit bridge.

The logic uses a small amount of dynamic power to wake up the C1-SME2 unit from low-power mode. Other than this power use, the drawn power is reduced to static leakage current only.

The C1-SME2 unit exits Standby mode when one of the following events occurs:

- The core detects a reset
- One of the cores sends arbitration request instructions or System register access to the C1-SME2 unit

4.2.2 Low-power mode behavior considerations

You must consider how certain events affect the low-power mode behavior of the C1-SME2 unit.

While the C1-SME2 unit is in Standby mode, the clocks in the C1-SME2 unit are temporarily enabled when any of the following events are detected:

- An access on the utility bus interface targeting the C1-SME2 unit or MSR/MRS to **IMPLEMENTATION DEFINED** System registers, Reliability, Availability, and Serviceability (RAS), or Maximum Power Mitigation Mechanism (MPMM)
- A debug access through the APB interface
- A system snoop request that must be serviced by the C1-SME2 unit L1 data cache
- A cache or Translation Lookaside Buffer (TLB) maintenance operation that must be serviced by the TLB
- A Distributed Virtual Memory (DVM) request
- A Pseudo-fault generation counter decrementing in the RAS registers

Note

The C1-SME2 unit does not exit Standby mode when the clocks are temporarily enabled.

4.3 Power control

The C1-DynamIQ™ Shared Unit (DSU) Power Policy Units (PPUs) control all power mode transitions for the C1-SME2 unit, the cores, and the cluster.

Each C1-SME2 unit has its own PPU to control its own unit power domain.

In addition, there is a PPU for the cluster.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

The PPUs decide and request any change in power mode. The C1-SME2 unit then performs any actions necessary to reach the requested power mode. For example, the unit might gate clocks, clean the L1 data cache, or disable coherency before it accepts the request.

For more information about the PPUs for the cluster, the cores and the unit, see the following sections in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*:

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

Related information

A.4.1 IMP_CMEPPMCR_EL3, Global PPM Configuration Register on page 161
A.2.2 IMP_CMEMPMMCR_EL3, Global MPMM Configuration Register on page 123
B.2.1 CMEPPMCR, Global PPM Configuration Register on page 227
B.2.2 CMEMPMMCR, Global MPMM Configuration Register on page 228

4.4 C1-SME2 unit power modes

The C1-SME2 unit power domain has a defined set of power modes and corresponding legal transitions between these modes. The power mode of each C1-SME2 unit can be independent of other C1-SME2 units in a cluster.

The Power Policy Unit (PPU) of a C1-SME2 unit manages at the cluster level the transitions between the power modes for that unit. For more information, see *Power management* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

The following table shows the supported C1-SME2 unit power modes.

Caution

Power modes that are not shown in the following table are not supported and must not occur. Deviating from the legal power modes can lead to **UNPREDICTABLE** results.

Table 4-1: C1-SME2 unit power modes

Power mode	Short name	Power state
On	ON	The C1-SME2 unit is powered up and active. **Note:** For general use, Arm® recommends using dynamic operating mode for the C1-SME2 unit PPU. This gives the most automation and quickest response times to requested power mode changes.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Power mode	Short name	Power state
Off	OFF	The C1-SME2 unit is powered down. **Caution:** The system deadlocks if all the following are true: • The C1-SME2 unit PPU is programmed in static mode • The C1-SME2 unit is in Off power mode • A core is in On power mode and tries to execute Streaming SVE (SSVE) instructions
Emulated Off	OFF_EMU	Emulated Off mode permits you to debug the powerup and powerdown cycle without changing the software. In this mode, the C1-SME2 unit proceeds through all the powerdown steps except when: • The clock is not gated and power is not removed when the C1-SME2 unit is powered down. • Only a Warm reset is asserted. All the RAM content is preserved in the C1-SME2 unit and remains accessible by the debugger.
Debug recovery	DBG_RECOV	The RAM and logic are powered up. This mode applies a Warm reset to the C1-DSU cluster, while preserving memory and Reliability, Availability, and Serviceability (RAS) registers for debug purposes. Both cache and RAS state are preserved when transitioning from DBG_RECOV to ON. **Caution:** This mode must not be used during normal system operation.
Warm reset	WARM_RST	A Warm reset resets all logic except for the RAS registers.

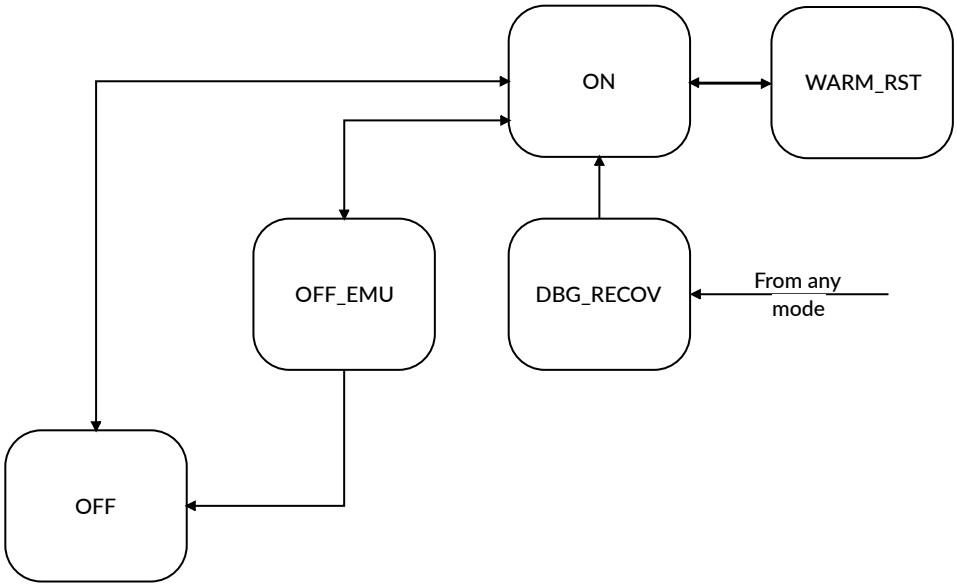
The following figure shows the supported modes for the C1-SME2 unit power domain and the legal transitions between them.

Note

- The automatic transition between On and Off power modes is possible only if the C1-SME2 unit PPU is in dynamic mode. This means that when the C1-SME2 unit PPU is in dynamic mode, no software support is required for the transition between On and Off power modes.
- Embedded Logic Analyzer (ELA) is not compatible with using C1-SME2 unit PPU in dynamic mode. To use ELA with the C1-SME2 unit, you need to program the C1-SME2 unit PPU in static On mode before programming the ELA. If the ELA registers are accessed while the C1-SME2 unit PPU is programmed in dynamic mode:
  - The ELA register values may be lost
  - Access to the ELA registers may return a SLVERR response

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 4-3: C1-SME2 unit power mode transitions



Related information

4.2 Architectural clock gating modes on page 32

4.4.1 On mode

In the On power mode, the C1-SME2 unit is on and fully operational.

To avoid system deadlock, the C1-SME2 Power Policy Unit (PPU) must be programmed in dynamic mode before a core executes the Streaming SVE (SSVE) instructions.

When the C1-SME2 PPU is programmed in dynamic mode, the C1-SME2 unit turns on automatically as soon as a core sends instructions to the C1-SME2 unit.

When a transition to the On mode is completed, the L1 data cache is accessible and coherent. No software support is required to turn on the C1-SME2 unit when the C1-SME2 PPU is in dynamic mode.

4.4.2 Off mode

In the Off power mode, power is removed completely from the C1-SME2 unit and no state is retained.

All the following conditions should be met for C1-SME2 to turn off automatically:

- The C1-SME2 Power Policy Unit (PPU) is programmed in dynamic mode
- All cores connected to C1-SME2 have PSTATE.SM and PSTATE.ZA set to 0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

- No interrupt is pending from the Reliability, Availability, and Serviceability (RAS) node

> **Note**
> If the RAS interrupts are enabled, and if RAS errors occur which cause RAS interrupts while the unit is powering down, then the RAS interrupts prevent the unit from powering down to prevent loss of information.

- No Pseudo-fault generation counter in the RAS registers is programmed to decrement a counter to generate an interrupt

Once these conditions are reached, C1-SME2 can start counting idle time and start turning off.

> **Caution**
> If the C1-SME2 PPU is programmed in static mode, and a core tries to execute Streaming SVE (SSVE) instructions when C1-SME2 is Off, the system deadlocks.

In Off mode, all C1-SME2 logic and RAMs are off. The values of **IMPLEMENTATION DEFINED** System registers, RAS, Maximum Power Mitigation Mechanism (MPMM), and Activity Monitoring Unit (AMU) registers are preserved and their values are restored when C1-SME2 is powered up. The L1 cache is disabled, cleaned, and invalidated. C1-SME2 is also removed from coherency automatically on transition to Off mode.

4.4.3 Emulated off mode

In Emulated off mode, all the RAM content is preserved. All other functional interfaces behave as if the C1-SME2 unit is in Off mode.

4.4.4 Debug recovery mode

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

By default, the C1-SME2 unit invalidates its cache when it transitions from Off to On mode. Using Debug recovery mode allows the L1 data cache content that was present before the reset to be observable after the reset. In this mode, the content of the L1 data cache is retained and is not altered on the transition back to the On mode.

In addition to preserving the cache content, Debug recovery supports preserving the Reliability, Availability, and Serviceability (RAS) state. A transition to Debug recovery mode is made from any state, which puts C1-SME2 into a Warm reset state. There is no external mechanism to apply a Warm reset mode other than programming the C1-DynamIQ™ Shared Unit (DSU) Power Policy Units (PPUs).

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Caution

Debug recovery is strictly for debug purposes and must not be used for functional purposes. When entering Debug recovery mode, correct operation of the L1 data cache is not guaranteed.

Debug recovery mode can occur at any time with no guarantee of the state of C1-SME2. A request of this type is accepted immediately, therefore its effects on C1-SME2, the C1-DSU 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. C1-SME2 does not expect these transactions to complete after a reset, which might cause a system deadlock.

If the system sends a snoop to the C1-DSU cluster during Debug recovery mode, and depending on the cluster state, the snoop:

- Might get a response and disturb the contents of the L1 data cache.
- Might not get a response and cause a system deadlock.

4.4.5 Warm reset mode

A Warm reset resets all states except for the Reliability, Availability, and Serviceability (RAS) registers.

Caution

WARM_RST mode is strictly for debug purposes.

A Warm reset is applied to the C1-SME2 unit when the unit Power Policy Unit (PPU) in the C1-DynamIQ™ Shared Unit (DSU) is programmed for WARM_RST mode.

WARM_RST mode is only expected to be used for resets triggered by a system level issue, such as a watchdog timeout.

WARM_RST mode can occur at any time with no guarantee of the state of the C1-SME2 unit. A request to transition to WARM_RST mode is accepted immediately. Therefore, its effects on the C1-SME2 unit, the core, the C1-DSU cluster, or the wider system are **UNPREDICTABLE**, and a wider system reset might be required. For example, if there were any outstanding memory transactions at the time of the reset, these transactions might complete after the reset. Unless the system interconnect is also reset, the cluster will not expect these transactions to complete after the reset, and a system deadlock might occur.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

4.5 Performance and power management

The C1-SME2 unit implements a Maximum Power Mitigation Mechanism (MPMM) that can be used to limit high activity events within the unit or trade off efficiency versus peak performance.

4.5.1 Maximum Power Mitigation Mechanism

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

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:

- Limits the number of units that can execute higher activity workloads
- Switches to a different Dynamic Voltage and Frequency Scaling (DVFS) operating point

Caution

MPMM must not be relied on as the only electrical safety mechanism. It is essentially a localized assistance mechanism that operates at C1-SME2 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. MPMM is a first line of defense rather than a complete solution.

Related information

A.4.1 IMP_CMEPPMCR_EL3, Global PPM Configuration Register on page 161
A.2.2 IMP_CMEMPMMCR_EL3, Global MPMM Configuration Register on page 123
B.2.1 CMEPPMCR, Global PPM Configuration Register on page 227
B.2.2 CMEMPMMCR, Global MPMM Configuration Register on page 228

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

5. Memory management

The C1-SME2 unit does not have a Memory Management Unit (MMU) and relies on the core MMU for address translation.

The core MMU handles the following for the C1-SME2 unit:

- For load/store instructions, the core sends the physical address, that is the address translation is already completed by the core MMU without the C1-SME2 unit requesting address translation.
- For the prefetcher, an address translation request is sent to the host core when needed. The prefetcher fetches in advance all the data that the C1-SME2 unit needs and stores it in the L1 data cache. The prefetcher operates in Virtual Address (VA) and requests address translation from the host core. Translation table entries are cached into a Translation Lookaside Buffer (TLB) in the prefetcher.

Any fault detected during the translation is handled inside the core.

For more information on memory management and address translation in the core, see the *Memory management* chapter in your core Technical Reference Manual (TRM).

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

5.1 Translation Lookaside Buffer entry content

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

Each TLB entry contains:

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

Each TLB entry is associated with both of the following:

- A particular Address Space Identifier (ASID) that allows context switches without requiring the TLB to be invalidated
- A Virtual Machine Identifier (VMID) that allows virtual machine switches by the hypervisor without requiring the TLB to be invalidated

5.2 Memory behavior and supported memory types

The C1-SME2 unit supports memory types defined in the Armv8-A architecture.

Device memory types have the following attributes:

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

G – Gathering

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

R – Reordering

The capability to reorder transactions

E – Early Write Acknowledgement

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

In Streaming SVE (SSVE) mode, all Device accesses are considered to have the GRE attributes set.

The following table shows how memory types are supported in C1-SME2, where the n prefix means the capability is not allowed.

Table 5-1: Supported memory types

Memory type	Shareability	Inner Cacheability	Outer Cacheability	Notes
Device nGnRnE	Outer Shareable	-	-	Treated as Device GRE
Device nGnRE	Outer Shareable[1]	-	-	Treated as Device GRE
Device nGRE	Outer Shareable[1]	-	-	Treated as Device GRE
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 5-2: Shareability for Normal memory on page 41.	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 C1-DynamIQ™ Shared Unit (DSU) is 0 (No Allocate)
Normal	See Table 5-2: Shareability for Normal memory on page 41.	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 C1-DSU is 1, therefore upgraded to Write and Read-Allocate

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

Table 5-2: Shareability for Normal memory

Shareability	Treated as
Non-shareable	Non-cacheable

[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.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Shareability	Treated as
Outer Shareable	Outer Shareable
Inner Shareable	Outer Shareable

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

6. L1 data memory system

The C1-SME2 unit L1 data memory system executes load and store instructions, and it services memory coherency requests. The L1 data memory system includes the L1 data cache, prefetcher, and Context storage.

The following table shows the L1 data memory system features.

Table 6-1: L1 data memory system features

Feature	Description
L1 data cache	• 64KB • 4-way set associative, 16 banks • Physically Indexed, Physically Tagged (PIPT) • Optionally protected with Error Correcting Code (ECC)
Cache line length	64 bytes
Cache policy	Pseudo Re-reference Interval Prediction (RRIP) cache replacement policy
Interface with vector execute and matrix execute	• 2×512-bit write paths and 2×512-bit read paths
Prefetcher	• Prefetcher indexed by instruction address • 128 streams
Context storage	6.3KB RAM per core in the cluster

6.1 L1 data cache behavior

The L1 data cache is invalidated automatically at reset unless the C1-SME2 unit power mode is initialized to Debug recovery mode.

In Debug recovery mode, the L1 data cache is not guaranteed to be functional and should not be enabled.

There is no operation to invalidate or clean the L1 data cache, which is fully managed in hardware. Cache maintenance operations (by physical or virtual address) executed in a core are propagated through the coherency system on all caches in the cluster, including the C1-SME2 L1 data cache.

Data Cacheability disabled behavior

If the data Cacheability is disabled, the following occurs:

- A new line is not allocated in the C1-DynamIQ™ Shared Unit (DSU) L3 cache as a result of a load instruction.
- All load and store instructions to cacheable memory are treated as Non-cacheable.

To maintain data coherency between two C1-SME2 units, the C1-SME2 unit uses the Modified Exclusive Shared Invalid (MESI) protocol.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Related information

4.4.4 Debug recovery mode on page 37

6.2 Write streaming capability

A cache line is allocated to the L1 data 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 gets overwritten by subsequent writes. In some situations, cache line allocation on writes is not required.

To prevent unnecessary cache line allocation, the prefetcher detects patterns and does not prefetch lines when it knows full lines will be produced.

6.3 Data prefetching

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

Preload instructions

For cases that cannot be handled efficiently by data prefetchers, the architecture defines the following instructions:

- Prefetch Memory instructions, `PRFM`
- Contiguous Prefetch instructions, `PRFB`, `PRFH`, `PRFW`, `PRFD`
- Range Prefetch Memory instructions, `RPRFM`

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

`PRFM` integer operation is changed to `NOP` that is, it is not executed by the C1-SME2 unit when in Streaming SVE (SSVE) mode.

The following Scalable Vector Extension (SVE) instructions are changed to `NOP` that is, they are not executed by the C1-SME2 unit when in SSVE mode:

- Contiguous prefetch bytes, `PRFB`
- Contiguous prefetch halfwords, `PRFH`
- Contiguous prefetch words, `PRFW`
- Contiguous prefetch doublewords, `PRFD`

`RPRFM` instruction is executed in the C1-SME2 unit. `RPRFM` instruction prefetches at most 1024 bytes from the base address within the memory region it describes.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

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

Hardware data prefetcher

In the L1 data memory system, the load/store unit includes a hardware prefetcher engine that is responsible for generating prefetches targeting L1 data cache and L3 cache. The hardware data prefetcher uses:

- The Virtual Address (VA)
- Part of the Program Counter (PC)
- Part of the instruction Opcode

The A.1.5 IMP_CMECFG_EL1, SME2 Configuration Register on page 102 allows control over some aspects of the prefetcher behavior, such as disabling the prefetcher.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

7. Direct access to internal memory

The C1-SME2 unit provides a mechanism to read the internal memory that the L1 cache uses through **IMPLEMENTATION DEFINED** System registers. You can use this mechanism to investigate any issues when the coherency between the cache data and the system memory data is broken.

Direct access to internal memory is available only in EL3. In all other exception levels, executing these instructions results in an Undefined Instruction exception.

You can access the contents of the internal memory using three read-only (RO) System registers in Table 7-1: System registers used to access internal memory on page 46. The internal memory is selected by using the following IMP_CMERAMINDEX System instruction:

```
SYS #6, C15, C12, #0, <Xt>
```

IMP_CMERAMINDEX, IMP_CMERAMINDEX System instruction reads data from the selected memory. The data is read from the read-only System registers as shown in the following table:

Note

- All the System registers are read-only (RO) and 64-bit wide
- For the register reset value, see the individual bit resets
- Any access to the data registers returns data. However, if IMP_CMERAMINDEX System instruction was not executed before or if it was executed by another core, data registers return zero.
- The direct cache access registers are available while the C1-SME2 unit is in the Off and Emulated off power modes. However, the direct cache accesses do not power on the unit or return valid cache contents. To return valid cache contents, you must ensure the unit is powered on.
- Click the register name for details on the returned data format

Table 7-1: System registers used to access internal memory

Register name	Description	Access encoding
IMP_CMERAMDATA0_EL3	Data0 for RAMINDEX	`MRS <Xt>, S3_6_C15_C12_2`
IMP_CMERAMDATA1_EL3	Data1 for RAMINDEX	`MRS <Xt>, S3_6_C15_C12_3`
IMP_CMERAMDATA2_EL3	Data2 for RAMINDEX	`MRS <Xt>, S3_6_C15_C12_4`

7.1 L1 data cache encodings

The L1 data cache is 4-way set associative and has a fixed size of 64KB.

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

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table 7-2: C1-SME2 L1 data cache tag location encoding

Bit field of Xn	Description
[63:56]	Reserved
[55:48]	Memory System Resource Partitioning and Monitoring (MPAM)
[47]	PF_SW, Line brought by SW prefetch
[46]	PF_HW, Line brought by HW prefetch
[45:44]	MEM_ATTR, Memory attributes
[43:40]	Page-Based Hardware Attributes (PBHA)
[39]	NS bit
[38]	Reserved
[37:35]	Reserved
[34:27]	Reserved
[26:0]	Physical address [39:13]

Table 7-3: C1-SME2 L1 data cache data location encoding for Data0

Bit field of Xn	Description
[63:0]	RAM Data range [63:0] Returns the data range [63:0] from a RAMINDEX instruction

Table 7-4: C1-SME2 L1 data cache data location encoding for Data1

Bit field of Xn	Description
[63:0]	RAM Data range [127:64] Returns the data range [127:64] from a RAMINDEX instruction

Table 7-5: C1-SME2 L1 data cache data location encoding for Data2

Bit field of Xn	Description
[63:12]	Reserved
[11:4]	Reserved
[3:0]	MTE_TAG, Data Memory Tagging Extension (MTE) Tag On a Cold reset, this field resets to an architecturally **UNKNOWN** value

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

8. RAS Extension support

The C1-SME2 unit supports the Reliability, Availability, and Serviceability (RAS) Extension, including all extensions up to Arm®v9.3-A.

In particular, the C1-SME2 unit supports these RAS Extension features:

- Fault Handling Interrupts (FHIs)
- Error Recovery Interrupts (ERIs)
- Poison attribute on bus transfers
- Global poison bit when the state is corrupted
- Cache protection with Single Error Correct, Double Error Detect (SECDED) and Error Correcting Code (ECC) on the functional RAMs that contain dirty data. This includes the L1 data cache tag, L1 data cache data, and Context storage RAMs.
- Error Record registers to help software perform recovery actions
- Error injection capabilities to facilitate software and system debug
- The Error Synchronization Barrier (ESB) instruction to synchronize unrecoverable errors. When an `ESB` instruction is executed, the unit ensures that all System Error (SError) interrupts that are generated by instructions before the `ESB` are either taken or deferred. If the unit cannot take the interrupt, it records the interrupt in the Deferred Interrupt Status Register DISR_EL1. For more information on DISR_EL1, see the *Arm® Architecture Reference Manual for A-profile architecture*.

Fault detection features are included in groups within the C1-DSU cluster, the cores, and the C1-SME2 unit. Each group of fault detection features is referred to as a node. You can access each node by using either the System registers or the utility bus.

The C1-SME2 unit has a RAS node per unit instance that is:

- Accessible through the core System register interfaces as a RAS node per unit instance
- Accessible through memory interface
- The node is per unit instance but shared by all the cores executing Streaming SVE (SSVE) mode instructions on this instance

For information on the nodes implemented in the C1-SME2 unit, the cores, and the C1-DSU cluster, see the *RAS extension support* chapter in your core Technical Reference Manual (TRM).

For more information on the architectural RAS Extension and the definition of a node, see the *Arm® Reliability, Availability, and Serviceability (RAS) System Architecture*

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

8.1 Cache protection behavior

The Reliability, Availability, and Serviceability (RAS) Extension that is implemented in the C1-SME2 unit includes cache protection. In this case, the C1-SME2 unit protects against errors that result in a RAM bitcell holding the incorrect value.

The C1-SME2 RAS implementation supports Single Error Correct, Double Error Detect (SECDED) and Error Correcting Code (ECC) on the RAMs that can hold dirty data. This includes L1 data cache tag, L1 data cache data, and Context storage RAMs. The data size is specific for each RAM and depends on the protection granule.

The C1-SME2 unit can progress and remain functionally correct when there is a single-bit error in any RAM.

If there are multiple single-bit errors in different RAMs, or within different protection granules within the same RAM, the C1-SME2 unit also remains functionally correct.

If there is a double-bit error in a single RAM within the same protection granule, the C1-SME2 unit 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.

If there are errors that are three or more bits within the same protection granule, the C1-SME2 unit might or might not detect the errors. Whether the C1-SME2 unit detects the errors or not depends on the RAM and the position of the errors within the RAM.

The cache protection feature of the C1-SME2 unit has no performance impact when no errors are present.

8.2 Error containment

The C1-SME2 unit supports error containment for data errors, which means detected data errors are not silently propagated. Data errors are deferred using data poisoning, ensuring that a consumer is aware of the error.

If any of the following occur when System Error (SError) is enabled, an error is reported to the core, which can trigger a SError:

- Poisoned data is consumed
- A register (GPR, Predicate, or Flags) is transferred to the core
- A partial store cannot be written to the L1 data cache and data is lost

Support for the Error Synchronization Barrier (`ESB`) instruction in the C1-SME2 unit allows further isolation of imprecise exceptions that are reported when a SError is reported.

An error detected inside the C1-SME2 unit RAMs is registered in the Reliability, Availability, and Serviceability (RAS) node.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Global poison bit and poison propagation

Most uncorrectable errors, including evictions, are deferred to a poison bit.

When the C1-SME2 unit detects a double error, or consumes a poisoned cache line, it sets a global poison bit indicating that the state has been corrupted. On a store, the error is propagated to the C1-SME2 L1 data cache, C1-DSU L3 cache, and the external memory.

An error is reported to the core when:

- The core consumes the memory data for which the poison bit has been propagated. In most cases, it is detected as a precise error.
- The C1-SME2 unit consumes corrupted data
- The C1-SME2 unit produces a Predicate register, GPR register, or PSTATE flags to the core. This is reported as an uncontainable System Error.

There is a Poison bit per core that can connect to the C1-SME2 unit.

An `ESB` instruction has no effect on the C1-SME2 poison bit, but ensures that any pending SError is synchronized before completion.

The global poison bit is cleared to 0 when one of the following conditions occurs:

- The core executes a `SMSTOP` instruction that clears both PSTATE.SM and PSTATE.ZA to 0
- The core takes a System Error (SError) when SError is enabled
- The core defers a System Error (SError) to DISR_EL1 when SError is not enabled

This has no effect on the poison bit stored in the C1-SME2 unit L1 data cache, core L1 cache and L2 cache, or shared C1-DynamIQ™ Shared Unit (DSU) L3 cache.

8.3 Fault detection and reporting

When the C1-SME2 unit 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 ERR0CTLR.FI is set, all detected Deferred errors, Uncorrected errors, and overflows of the corrected error counters generate an FHI. When ERR0CTLR.CFI is set, all detected Corrected errors also generate an FHI.

FHIs from C1-SME2 unit m are signaled using nCOREFAULTIRQ[m+NUM_CORES], where NUM_CORES is the number of cores in the cluster.

Error recovery interrupts

When ERR0CTLR.UI is set, all detected Uncorrected errors that are not deferred generate an ERI.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

ERIs from C1-SME2 unit m are signaled using nCOREERRIRQ[m+NUM_CORES], where NUM_CORES is the number of cores in the cluster.

8.4 Error injection

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

Error injection uses the error detection and reporting registers to insert errors. The C1-SME2 unit 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 data cache, or as a result of a snoop on the L1 data cache.

Uncontainable errors

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

The C1-SME2 unit supports the ERXPFGF_EL1.NA bit, meaning that error and fault injections can be generated without performing a memory access. For more information, see A.3.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature Register on page 144.

An error can be injected immediately or when a 32-bit counter reaches zero. You can control the value of the counter through the Error Pseudo-fault Generation Countdown Register, ERXPFGCDN_EL1. The value of the counter decrements on a per clock cycle basis. For more information about ERXPFGCDN_EL1, see the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

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

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

8.5 AArch64 RAS registers

The following summary table provides an overview of **IMPLEMENTATION DEFINED** RAS registers in the core.

Table 8-1: RAS registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ERRIDR_EL1	3	0	C5	C3	0	See individual bit resets.	64-bit	Error Record ID Register
ERRSELR_EL1	3	0	C5	C3	1	See individual bit resets.	64-bit	Error Record Select Register
ERXFR_EL1	3	0	C5	C4	0	See individual bit resets.	64-bit	Selected Error Record Feature Register
ERXCTLR_EL1	3	0	C5	C4	1	See individual bit resets.	64-bit	Selected Error Record Control Register
ERXSTATUS_EL1	3	0	C5	C4	2	See individual bit resets.	64-bit	Selected Error Record Primary Status Register
ERXADDR_EL1	3	0	C5	C4	3	See individual bit resets.	64-bit	Selected Error Record Address Register
ERXPFGF_EL1	3	0	C5	C4	4	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Feature Register
ERXPFGCTL_EL1	3	0	C5	C4	5	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Control Register
ERXPFGCDN_EL1	3	0	C5	C4	6	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Countdown Register
ERXMISC0_EL1	3	0	C5	C5	0	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 0
ERXMISC1_EL1	3	0	C5	C5	1	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 1
ERXMISC2_EL1	3	0	C5	C5	2	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 2
ERXMISC3_EL1	3	0	C5	C5	3	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 3

8.6 External RAS registers

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped RAS registers in the core.

Table 8-2: RAS registers summary

Offset	Name	Reset	Width	Description
0x0	ERR0FR	`0x005100008010A9A2`	64-bit	Error Record <n> Feature Register
0x8	ERR0CTLR	See individual bit resets.	64-bit	Error Record <n> Control Register
0x10	ERR0STATUS	See individual bit resets.	64-bit	Error Record <n> Primary Status Register
0x18	ERR0ADDR	See individual bit resets.	64-bit	Error Record <n> Address Register

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Offset	Name	Reset	Width	Description
0x20	ERR0MISC0	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 0
0x28	ERR0MISC1	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 1
0x30	ERR0MISC2	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 2
0x38	ERR0MISC3	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 3
0x800	ERR0PFGF	`0x0000000070001A63`	64-bit	Error Record <n> Pseudo-fault Generation Feature Register
0x808	ERR0PFGCTL	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Control Register
0x810	ERR0PFGCDN	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Countdown Register
0xE00	ERRGSR	See individual bit resets.	64-bit	Error Group Status Register
0xE10	ERRIIDR	`0xD8D1243B`	32-bit	Implementation Identification Register
0xFA8	ERRDEVAFF	See individual bit resets.	64-bit	Device Affinity Register
0xFBC	ERRDEVARCH	`0x47710A00`	32-bit	Device Architecture Register
0xFC8	ERRDEVID	`0x00000001`	32-bit	Device Configuration Register
0xFD0	ERRPIDR4	`0x00000004`	32-bit	Peripheral Identification Register 4
0xFE0	ERRPIDR0	`0x0000008D`	32-bit	Peripheral Identification Register 0
0xFE4	ERRPIDR1	`0x000000BD`	32-bit	Peripheral Identification Register 1
0xFE8	ERRPIDR2	`0x0000001B`	32-bit	Peripheral Identification Register 2
0xFEC	ERRPIDR3	`0x00000020`	32-bit	Peripheral Identification Register 3
0xFF0	ERRCIDR0	`0x0000000D`	32-bit	Component Identification Register 0
0xFF4	ERRCIDR1	`0x000000F0`	32-bit	Component Identification Register 1
0xFF8	ERRCIDR2	`0x00000005`	32-bit	Component Identification Register 2
0xFFC	ERRCIDR3	`0x000000B1`	32-bit	Component Identification Register 3

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

9. Utility bus

The utility bus provides access to control registers for various system components in the C1-DynamIQ™ Shared Unit (DSU), the cores, and the C1-SME2 unit within the C1-DSU cluster. The utility bus is implemented as a 64-bit AMBA AXI5 subordinate port, and the control registers are memory-mapped onto the utility bus.

The utility bus provides access to the following system functions in the C1-SME2 unit:

- Reliability, Availability, and Serviceability (RAS) registers for the unit
- Activity Monitor Unit (AMU) registers in the unit
- Max Power Mitigation Mechanism (MPMM) registers in the unit

---

Note

Information about the Power Policy Unit (PPU) registers for the C1-SME2 unit in the cluster is provided in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*. For all other registers accessed by the utility bus, see *Utility bus* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

9.1 Base addresses for system components

Each set of System registers is grouped on separate 64KB page boundaries allowing access to be enforced by the Memory Management Unit (MMU) of the cores.

The following table shows the base addresses for each set of system component registers and what Security state they should be accessed from.

Note

- The base address for each set of registers for the C1-SME2 unit RAS, AMU, and MPMM registers depend on <m> and NUM_CORES where:
  - <m> is the C1-SME2 unit instance number in the cluster. <m> has a value of either 0 or 1
  - NUM_CORES is the total number of the cores in the cluster
- In the following table, any address space that is not documented is treated as **RAZ/WI**
- The base addresses in the following table are the addresses accessed on the utility bus interface. The system interconnect typically maps these addresses into a particular address range based on the system address map. Therefore, software has to add the base address listed here onto the system address range base to get the absolute physical address of a register.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table 9-1: Utility bus base addresses for system component registers

Base address	Registers	Security state	Memory map
`0x<m+NUM_CORES>9_0000`	SME2 <m> AMU	Secure Secure	B.1 External AMU registers summary on page 171
`0x<m+NUM_CORES>A_0000`	SME2 <m> RAS	Secure	A.3 AArch64 RAS registers summary on page 130
`0x<m+NUM_CORES>B_0000`	SME2 <m> MPMM	Secure	B.2 External MPMM registers summary on page 226
`0x<m+NUM_CORES>D_0000` - `0x<m+NUM_CORES>F_0000`	Reserved	-	-

Note

For more information on utility bus base addresses for system component registers, see *Utility bus* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

10. System control

The System registers control and provide status information for the functions that the C1-SME2 unit implements.

The main functions of the System registers are:

- Overall system control and configuration
- C1-SME2 unit control and configuration
- CME Assignment Block (CAB) control and configuration

The System registers are accessible in AArch64 Execution state at EL1 to EL3.

10.1 Accessing the System registers

Each C1-SME2 unit instance has its own set of System registers. You must select the C1-SME2 unit instance before accessing the System registers.

Accessing the C1-SME2 unit IMPLEMENTATION DEFINED registers

Before accessing an **IMPLEMENTATION DEFINED** system register, program the A.1.2 IMP_CMESELR_EL1, SME2 Selection Register on page 95 with the correct instance index by using the following sequence. This is required even when a single C1-SME2 unit is implemented.

```
mov     x0, #<cme instance>
msr     S3_0_C15_C11_1, x0          // IMP_CMESELR_EL1: <cme_instance>
isb
<Access to IMPLEMENTATION DEFINED register>
```

The number of the C1-SME2 units implemented is accessible in the A.1.1 IMP_CMECFR_EL1, SME2 Features Register on page 93. The following sequence returns the number of the C1-SME2 unit instances in the cluster to x0:

```
mov     x0, #0
msr     S3_0_C15_C11_1, x0          // IMP_CMESELR_EL1
isb
mrs     x0, S3_0_C15_C11_0          // IMP_CMECFR_EL1
ubfx    x0, x0, #32, #4
```

All C1-SME2 instances must be programmed with the same values when two C1-SME2 instances are present in the cluster.

```
mov     x0, #0
msr     S3_0_C15_C11_1, x0          // IMP_CMESELR_EL1
isb
... Program CME0 registers

mov     x1, #1
msr     S3_0_C15_C11_1, x1          // IMP_CMESELR_EL1
isb
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
... Program CME1 registers in the same way
```

Note

- The A.1.2 IMP_CMESELR_EL1, SME2 Selection Register on page 95 needs to be programmed to execute the IMP_CMERAMINDEX System instruction and read the IMP_CMERAMDATAx_EL3 registers.
- When the C1-SME2 unit is powered off, accesses to the unit system registers are available and complete successfully. The accesses do not power on the unit, and it is not required to execute SMSTART before using the unit system registers.

Accessing the C1-SME2 unit RAS registers

Each C1-SME2 unit contains a Reliability, Availability, and Serviceability (RAS) node. The RAS node is accessed by A.3.2 ERRSELR_EL1, Error Record Select Register on page 133. For more information, see 8. RAS Extension support on page 48.

Accessing the C1-SME2 unit CAB registers

CME Assignment Block (CAB) registers are not located in the C1-SME2 unit and do not need to be connected to the C1-SME2 unit. There is no need to be connected to C1-SME2 to access CAB registers.

10.2 Arbitration control

The C1-SME2 unit is shared between all the cores inside a C1-DSU cluster.

When several cores try to access the same C1-SME2 unit instance, the C1-SME2 unit arbitrates between the requesting cores. The following fields of the A.1.5 IMP_CMECFG_EL1, SME2 Configuration Register on page 102 impact the arbitration behavior:

FAIRSHARE_UPPER

Defines the split of Exclusive priorities and Fair-share priorities. For example, if FAIRSHARE_UPPER has a value of 14, priorities 0 to 14 use Fair-share priorities, and priority 15 uses an Exclusive priority.

Only the Fair-share priorities guarantee forward progress. Forward progress for cores using Exclusive priorities can only be guaranteed by the software. For more information, sSee 1.1.1 Streaming execution priority on page 12.

TIMEOUT_SWITCH, PRIORITY_RATIO

For Fair-share priorities, TIMEOUT_SWITCH defines the maximum number of cycles for Priority 0 before switching to the next requesting core. It is ignored if no other core requested arbitration. For example, TIMEOUT_SWITCH value of 0, indicates 512 cycles for Priority 0.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

A small value for TIMEOUT_SWITCH lowers the latency to access C1-SME2 unit, but it can impact the global efficiency for the following reasons:

- Switch time between two cores is typically in the range of 100 to 200 cycles
- As the caches have been filled for one core, switching to another core means that prefetcher needs to be restarted to fetch correct data

PRIORITY_RATIO defines the time allocation in cycles from one Fair-share priority to the next one. PRIORITY_RATIO adds 128 (¼), 256 (½) or 512 (1) cycles per core priority level. The following formula shows the added cycles in relation to the core priority level, PRIORITY_RATIO, and TIMEOUT_SWITCH:

```
512 x (1 + core priority level x PRIORITY_RATIO) x 2^(TIMEOUT_SWITCH)
```

For example, for TIMEOUT_SWITCH value of 0:

- Core priority level 0: 512 cycles
- Core priority level 1:
  - 512 + 128 cycles (for PRIORITY_RATIO ¼)
  - 512 + 256 cycles (for PRIORITY_RATIO ½)
  - 512 + 512 cycles (for PRIORITY_RATIO 1)
- Core priority level 2:
  - 512 + 2x128 cycles (for PRIORITY_RATIO ¼)
  - 512 + 2x256 cycles (for PRIORITY_RATIO ½)
  - 512 + 2x512 cycles (for PRIORITY_RATIO 1)

TIMEOUT_IDLE, TIMEOUT_SMSTOP, TIMEOUT_KEEP_ARB

The values TIMEOUT_IDLE and TIMEOUT_SMSTOP can be used to ensure a core that is not using the C1-SME2 unit loses arbitration when the either of the following conditions occur:

- TIMEOUT_IDLE triggers when a core stops sending instruction, independently of the value of PSTATE.SM or PSTATE.ZA
- TIMEOUT_SMSTOP triggers after PSTATE.SM is set to 0, independently of PSTATE.ZA value

When either of the two previous conditions occur, the arbitration is provided to another core.

Value TIMEOUT_KEEP_ARB can be used to define a minimum number of cycles to allocate to a core before it can switch to another core. This is used even when:

- Another core requests arbitration with Exclusive priority
- The arbitrated core becomes idle or PSTATE.SM is set to 0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

10.3 Clock gate and power control

The C1-SME2 unit is able to power on and off when the Power Policy Unit (PPU) is programmed in dynamic mode.

The following fields of the A.1.6 IMP_CMEPWR_EL1, SME2 Power Register on page 106 control the clock gate and power-off sequence:

TIMEOUT_TO_PWRDWN

Defines the number of system clock cycles before the C1-SME2 unit goes to low-power mode, causing the clocks to be gated. The counter starts counting when the unit is fully idle, including execution of instructions and memory transactions.

Setting a small value has a small impact on performance, as the number of cycles to leave this mode is typically in the range of 10-20 cycles. Setting a high value means that less power saving is achieved.

TIMEOUT_TO_CLKGATE

Defines the number of cycles counted by virtual counter CNTVCT_EL0 when the C1-SME2 unit is already in lower-power mode that is the clocks are gated before it starts power-down sequence.

This value is ignored if the conditions to enter low-power mode are not met.

Setting a small value can impact the performance for the following reasons:

- The C1-SME2 unit flushes the cache during the powers-off transition. If a new instruction or arbitration request is received during this time, the process can be abandoned but the core will need to fill data into the cache.
- Once the C1-SME2 unit is off, the power-up time of the C1-SME2 unit itself is in the range of 500-1000 cycles, but additional time could be dependent on the system, for example, turning on the power delivery network.

For more information, see 4.4 C1-SME2 unit power modes on page 34.

10.4 AArch64 Generic System Control registers

The following summary table provides an overview of **IMPLEMENTATION DEFINED** Generic System Control registers in the core.

Table 10-1: Generic System Control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CMECFR_EL1	3	0	C15	C11	0	See individual bit resets.	64-bit	SME2 Features Register
IMP_CMESELR_EL1	3	0	C15	C11	1	See individual bit resets.	64-bit	SME2 Selection Register
IMP_CMEACTLR_EL1	3	0	C15	C11	2	See individual bit resets.	64-bit	SME2 Auxiliary Control Register
IMP_CMEACTLR2_EL1	3	0	C15	C11	3	See individual bit resets.	64-bit	SME2 Auxiliary Control Register 2
IMP_CMECFG_EL1	3	0	C15	C11	4	See individual bit resets.	64-bit	SME2 Configuration Register
IMP_CMEPWR_EL1	3	0	C15	C11	5	See individual bit resets.	64-bit	SME2 Power Register

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CMEREVIDR_EL1	3	0	C15	C11	6	`0x0000000000000000`	64-bit	SME2 unit ECO ID Register
IMP_CABACTLR_EL1	3	0	C15	C12	5	See individual bit resets.	64-bit	CAB Auxiliary Control Register
IMP_CABECTLR_EL1	3	0	C15	C12	6	See individual bit resets.	64-bit	CAB Control Register
IMP_CMERAMDATA0_EL3	3	6	C15	C12	2	See individual bit resets.	64-bit	Data0 for RAMINDEX
IMP_CMERAMDATA1_EL3	3	6	C15	C12	3	See individual bit resets.	64-bit	Data1 for RAMINDEX
IMP_CMERAMDATA2_EL3	3	6	C15	C12	4	See individual bit resets.	64-bit	Data2 for RAMINDEX

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

11. Debug

The C1-SME2 unit does not have debug logic. The C1-SME2 unit supports debug through the core it is connected to.

For more information on Debug for the core, see the *Debug* chapter in your core Technical Reference Manual (TRM).

The C1-DSU 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 C1-DSU cluster, and others in the separate DebugBlock.

The DebugBlock is a dedicated debug component in the C1-DSU, 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 C1-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.

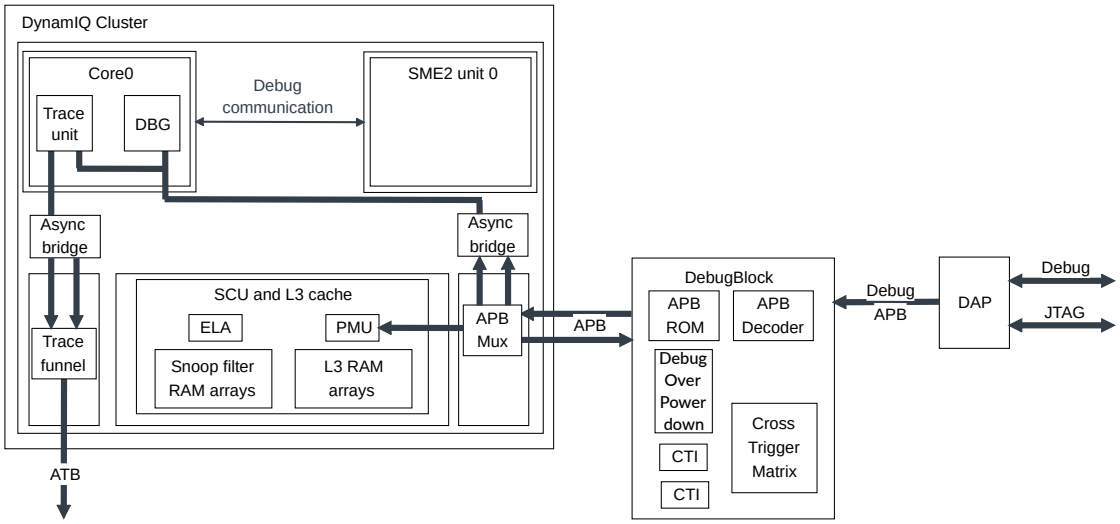
The debug system implements the following CoreSight debug components:

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

The following figure shows how the debug system is implemented with the C1-DSU cluster.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 11-1: C1-DSU cluster debug components



For more information about the C1-DSU cluster debug components, see the *Debug* chapter in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

The C1-SME2 unit 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 cache uses. For more information, see 7. Direct access to internal memory on page 46.

11.1 Debug register interfaces

The C1-SME2 unit does not have Debug register interfaces. Debug for the C1-SME2 unit is done through the core it is connected to.

You can access the C1-SME2 unit in debug state through the core debug interfaces.

For more information, see the *Debug register interfaces* section in the *Debug* chapter of your core *Technical Reference Manual* (TRM).

11.1.1 C1-SME2 unit interfaces

You can access the Embedded Logic Analyzer (ELA) registers from the C1-SME2 unit using the Advanced Peripheral Bus (APB) completer port that connects into the DebugBlock of the C1-DynamIQ™ Shared Unit (DSU).

The ELA-600 is licensed separately, and the ELA registers are still present even if the ELA configuration parameter indicates that support for the ELA is not included.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

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

For information on APB completer port interface, see the *Debug* chapter or the *Interfaces* section in the *Technical overview* chapter of the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

11.2 Debug memory map and debug signals

The debug memory map and debug signals are handled at the C1-DSU cluster level.

For more information, see *Debug* and *ROM tables* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

11.3 ROM table

The C1-SME2 unit includes a ROM table that contains a list of components in the system. Debuggers must use the ROM table to determine which CoreSight™ components are implemented.

The ROM table is a CoreSight™ debug related component that aids system debug along with CoreSight™ SoC. There is one ROM table for each core, complex, and C1-SME2 unit. ROM tables comply with the *Arm® CoreSight™ Architecture Specification v3.0*.

The C1-DynamIQ™ Shared Unit (DSU) 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, complex, and C1-SME2 unit. For more information, see *ROM tables* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

The C1-SME2 unit ROM table includes the following entries:

Table 11-1: C1-SME2 unit ROM table

Offset	Name	Description
`0x0000`	ROMENTRY0	C1-SME2 unit Embedded Logic Analyzer (ELA), ELA-600 debug (when ELA-600 present)

Related information

B.4 External ROM table registers summary on page 296

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

11.4 CoreSight component identification

Each component associated with the C1-SME2 unit has a unique set of CoreSight™ ID values. The following table shows these values.

Table 11-2: C1-SME2 unit CoreSight™ component identification

Component	Peripheral ID	Component ID	DevType	DevArch	Unit revision
ROM table	`0x04201BBD8D`	`0xB105900D`	`0x00`	`0x47700AF7`	r1p2
AMU	`0x04201BBD8D`	`0xB105900D`	`0x16`	`0x47700A66`	r1p2
RAS	`0x04201BBD8D`	`0xB105F00D`	`0x00`	`0x47710A00`	r1p2

11.5 External ROM table registers

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped ROM table registers in the core.

Table 11-3: ROM table registers summary

Offset	Name	Reset	Width	Description
0x0	ROMENTRY0	See individual bit resets.	32-bit	Class 0x9 ROM Table Entries
0xFB8	AUTHSTATUS	See individual bit resets.	32-bit	Authentication Status Register
0xFBC	DEVARCH	`0x47700AF7`	32-bit	Device Architecture Register
0xFC8	DEVID	See individual bit resets.	32-bit	Device Configuration Register
0xFD0	PIDR4	`0x00000004`	32-bit	Peripheral Identification Register 4
0xFE0	PIDR0	`0x0000008D`	32-bit	Peripheral Identification Register 0
0xFE4	PIDR1	`0x000000BD`	32-bit	Peripheral Identification Register 1
0xFE8	PIDR2	`0x0000001B`	32-bit	Peripheral Identification Register 2
0xFEC	PIDR3	`0x00000020`	32-bit	Peripheral Identification Register 3
0xFF0	CIDR0	`0x0000000D`	32-bit	Component Identification Register 0
0xFF4	CIDR1	`0x00000090`	32-bit	Component Identification Register 1
0xFF8	CIDR2	`0x00000005`	32-bit	Component Identification Register 2
0xFFC	CIDR3	`0x000000B1`	32-bit	Component Identification Register 3

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

12. Performance Monitors Extension support

The C1-SME2 unit implements the Performance Monitors Extension, including support for performance monitoring features up to Arm®v8.8-A.

The C1-SME2 unit Performance Monitoring Unit (PMU):

- Collects events through an event interface from other blocks in the C1-SME2 unit. These events are used as triggers for event counters.
- Reports these events to the core as they are counted in the core PMU counters.
- Provides up to 31 PMU 64-bit counters within the cores that count any of the events available in the C1-SME2 unit. 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 cannot program the PMU counters in the C1-SME2 unit directly. However, you can program the PMU counters in the core to select the performance events relevant for the C1-SME2 unit.

For guidance on collecting and analyzing Arm® C1-Scalable Matrix Extension 2 telemetry data, see the *Arm® C1-Scalable Matrix Extension 2 Telemetry Specification*.

12.1 Common performance monitoring unit events

The C1-SME2 unit Performance Monitoring Unit (PMU) collects events from other units in the design and uses numbers to reference these events.

Common PMU events

The following table shows the C1-SME2 unit performance monitors events that are generated and the numbers that the PMU uses to reference the events. The table also shows the bit position of each event on the event bus. Event numbers that are not listed are reserved.

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

Note

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

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table 12-1: Common PMU events

Event number	Mnemonic	Description
0x1B	INST_SPEC	**Operation speculatively executed** Counts instructions that have been speculatively executed.
0x74	ASE_SPEC	**Operation speculatively executed, Advanced SIMD** The counter counts each operation counted by INST_SPEC that is an Advanced SIMD data-processing operation. It does not count SVE operations counted by SVE_SPEC, or SME operations counted by SME_SPEC.
0x8	INST_RETIRED	**Operation retired** Counts instructions that have been architecturally executed.
0x8000	SIMD_INST_RETIRED	**Operation retired, SIMD, including load and store** The counter counts each retired SIMD instruction counted by INST_RETIRED. This counter does not count scalar instructions counted by ASE_INST_RETIRED.
0x8004	SIMD_INST_SPEC	**Operation speculatively executed, SIMD, including load and store** The counter counts each speculatively executed SIMD operation counted by INST_SPEC. This counter does not count scalar operations counted by ASE_INST_SPEC.
0x8010	FP_SPEC	**Floating-point operation speculatively executed, including SIMD** The counter counts each Speculatively executed floating-point operation due to an A64 scalar, Advanced SIMD, SVE or SME instruction listed in SVE floating-point instructions.
0x8014	FP_HP_SPEC	**Floating-point operation speculatively executed, half precision** Counts speculatively executed half precision floating point operations.
0x8018	FP_SP_SPEC	**Floating-point operation speculatively executed, single precision** Counts speculatively executed single-precision floating point operations.
0x801C	FP_DP_SPEC	**Floating-point operation speculatively executed, double precision** Counts speculatively executed double-precision floating point operations.
0x8040	INT_SPEC	**Integer Operation speculatively executed** Counts integer operations that have been speculatively executed.
0x8043	ASE_SVE_INT_SPEC	**Operation speculatively executed, Advanced SIMD or SVE integer** The counter counts each Advanced SIMD or SVE operation counted by INT_SPEC where the type is integer.
0x8056	SVE_SPEC	**Operation speculatively executed, SVE** The counter counts each operation counted by ASE_SVE_SPEC that is a scalable vector data processing operation. It does not count: • SME operations counted by SME_SPEC. • Neon operation counted by ASE_SPEC • Load/store operations

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x8057	ASE_SVE_SPEC	**Operation speculatively executed, Advanced SIMD or SVE** The counter counts each operation counted by SE_SPEC that is an Advanced SIMD or scalable vector data processing operation. It does not count: • SME operations counted by SME_SPEC. • Load/store operations See ASE_SPEC and SVE_SPEC for these classifications.
0x8074	SVE_PRED_SPEC	**Operation speculatively executed, SVE predicated** The counter counts each Speculatively executed SIMD data-processing, load, or store operation due to an SVE instruction with a Governing predicate operand that determines the Active elements. Note: this counts SME operations requiring PSTATE.ZA to be set, including 2D operations.
0x8075	SVE_PRED_EMPTY_SPEC	**Operation speculatively executed, SVE predicated with no active predicates** The counter counts each Speculatively executed predicated SIMD data-processing, load, or store operation counted by SVE_PRED_SPEC where all elements are Inactive. That is, all elements in the Governing predicate are FALSE.
0x8076	SVE_PRED_FULL_SPEC	**Operation speculatively executed, SVE predicated with all active predicates** The counter counts each Speculatively executed predicated SIMD data-processing, load, or store operation counted by SVE_PRED_SPEC where all elements are Active. That is, all elements in the Governing predicate are TRUE.
0x8077	SVE_PRED_PARTIAL_SPEC	**Operation speculatively executed, SVE predicated with partially active predicates** The counter counts each Speculatively executed predicated SIMD data-processing, load, or store operation counted by SVE_PRED_SPEC that is not counted by either SVE_PRED_EMPTY_SPEC or SVE_PRED_FULL_SPEC. That is, the elements in the Governing predicate are neither all TRUE nor all FALSE.
0x8078	SVE_UNPRED_SPEC	**Operation speculatively executed, SVE unpredicated** The counter counts each Speculatively executed SIMD data-processing, load, or store operation due to an SVE instruction without a Governing predicate operand. This counts the SME operations requiring PSTATE.ZA to be set. It does not count Advanced SIMD operations.
0x8079	SVE_PRED_NOT_FULL_SPEC	**Operation speculatively executed, SVE predicated with no active or partially active predicates** Counts speculatively executed predicated SVE operations with at least one non active predicate elements.
0x8080	SVE_LDST_SPEC	**Operation speculatively executed, SVE load, store, or prefetch** The counter counts each Speculatively executed load or store operation counted by any of SVE_LD_SPEC or SVE_ST_SPEC. This includes Load/Store operations to Z register but does not count Load/Store operations to ZT and ZA registers.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x8081	SVE_LD_SPEC	**Operation speculatively executed, SVE load** The counter counts each Speculatively executed operation that reads from memory due to an SVE load instruction.
0x8082	SVE_ST_SPEC	**Operation speculatively executed, SVE store** The counter counts each Speculatively executed operation that writes to memory due to an SVE store instruction.
0x80C0	FP_SCALE_OPS_SPEC	**Scalable floating-point element ALU operations speculatively executed** Counts speculatively executed scalable floating point ALU operations.
0x80D0	FP_SCALE2_OPS_SPEC	**Scalable floating-point matrix element arithmetic operations speculatively executed** The counter counts each speculatively executed scalable floating-point matrix arithmetic operation counted by FP_SPEC. The counter increments by the number of numerical operations carried out by the instruction per 128/element_size by 128/element_size tile of the result.
0x80D2	FP_HP_SCALE2_OPS_SPEC	**Scalable floating-point half-precision matrix element arithmetic operations speculatively executed** The counter counts each Streaming SVE half-precision floating-point matrix ALU operation counted by `SME_FP_SPEC` that was speculatively executed. The counter increments by the number of numerical operations carried out by the instruction per 128/element_size by 128/element_size tile of the result.
0x80D3	FP_BF16_SCALE2_OPS_SPEC	**Scalable floating-point BFloat16 matrix element arithmetic operations speculatively executed** The counter counts each Streaming SVE BFloat16 floating-point matrix ALU operation counted by `SME_FP_SPEC` that was speculatively executed. The counter increments by the number of numerical operations carried out by the instruction per 128/element_size by 128/element_size tile of the result.
0x80D4	FP_SP_SCALE2_OPS_SPEC	**Scalable floating-point single-precision matrix element arithmetic operations speculatively executed** The counter counts each Streaming SVE single-precision floating-point matrix ALU operation counted by `SME_FP_SPEC` that was speculatively executed. The counter increments by the number of numerical operations carried out by the instruction per 128/element_size by 128/element_size tile of the result.
0x80D6	FP_DP_SCALE2_OPS_SPEC	**Scalable floating-point double-precision matrix element arithmetic operations speculatively executed** The counter counts each Streaming SVE double-precision floating-point matrix ALU operation counted by `SME_FP_SPEC` that was speculatively executed. The counter increments by the number of numerical operations carried out by the instruction per 128/element_size by 128/element_size tile of the result.
0x80E3	ASE_SVE_INT8_SPEC	**Operation speculatively executed, Advanced SIMD or SVE 8-bit integer** The counter counts each operation counted by ASE_SVE_INT_SPEC where the largest type is 8-bit integer.
0x80E7	ASE_SVE_INT16_SPEC	**Operation speculatively executed, Advanced SIMD or SVE 16-bit integer** The counter counts each operation counted by ASE_SVE_INT_SPEC where the largest type is 16-bit integer.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x80EB	ASE_SVE_INT32_SPEC	**Operation speculatively executed, Advanced SIMD or SVE 32-bit integer** The counter counts each operation counted by ASE_SVE_INT_SPEC where the largest type is 32-bit integer.
0x80EF	ASE_SVE_INT64_SPEC	**Operation speculatively executed, Advanced SIMD or SVE 64-bit integer** The counter counts each operation counted by ASE_SVE_INT_SPEC where the largest type is 64-bit integer.
0x8352	SME_FP_SPEC	**Operation speculatively executed, SME floating-point** The counter counts each speculatively executed floating-point operation counted by SE_SPEC due to an instruction which reads from or writes to any part of the ZA array.
0x835C	SME_SPEC	**Operation speculatively executed, SME data processing** The counter counts each operation counted by SE_SPEC that is an SME data-processing operation. Operations due to the following instructions are counted as SME data-processing operations: • Data-processing operations involving the ZA and ZT registers. Operations due to instructions added by FEAT_SME which involve the SVE registers but do not involve any ZA or ZT registers are counted as SVE data-processing operations.
0x835D	SE_SPEC	**Operation speculatively executed, Advanced SIMD, scalable vector extension, or scalable matrix extension data processing** The counter counts each operation counted by INST_SPEC that is an Advanced SIMD, scalable vector extension, or scalable matrix extension data-processing operation. See ASE_SVE_SPEC and SME_SPEC for these classifications.
0x835E	SME_INST_SPEC	**Operation speculatively executed, SME** The counter counts each speculatively executed operation counted by SE_INST_SPEC that is classified as an SME operation. Operations due to the following instructions are counted as SME operations: • Data-processing operations involving the ZA and ZT registers. • Load and store operations involving the ZA and ZT registers. Operations due to instructions added by FEAT_SME which involve the SVE registers but do not involve any ZA or ZT registers are counted as SVE data-processing operations.
0x8360	SME_INT8_SPEC	**Operation speculatively executed, SME 8-bit integer** The counter counts each speculatively executed 8-bit integer operation counted by SME_INT_SPEC due to an instruction which reads from or writes to any part of the ZA or ZT array.
0x8362	SME_FP_BF16_SPEC	**Operation speculatively executed, SME BFloat16 floating-point** The counter counts each speculatively executed BFloat16 floating-point operation counted by SME_FP_SPEC due to an instruction which reads from or writes to any part of the ZA array.
0x8364	SME_INT16_SPEC	**Operation speculatively executed, SME 16-bit integer** The counter counts each speculatively executed 16-bit integer operation counted by SME_INT_SPEC due to an instruction which reads from or writes to any part of the ZA or ZT array.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x8366	SME_FP_HP_SPEC	**Operation speculatively executed, SME half-precision floating-point** The counter counts each speculatively executed half-precision floating-point operation counted by SME_FP_SPEC due to an instruction which reads from or writes to any part of the ZA array.
0x836A	SME_FP_SP_SPEC	**Operation speculatively executed, SME single-precision floating-point** The counter counts each speculatively executed single-precision floating-point operation counted by SME_FP_SPEC due to an instruction which reads from or writes to any part of the ZA array.
0x8370	SME_FP_ADDSUB_SPEC	**Operation speculatively executed, SME floating-point addition or subtraction** The counter counts each speculatively executed floating-point addition or subtraction operation counted by SME_FP_SPEC due to an instruction which reads from or writes to any part of the ZA array.
0x8372	SME_FP_FMA_SPEC	**Operation speculatively executed, SME floating-point multiply-add or multiply-subtract** The counter counts each speculatively executed floating-point multi-vector multiply and accumulate, or multi-vector multiply and subtract instruction counted by SME_FP_SPEC due to an operation which reads from or writes to any part of the ZA array.
0x8374	SME_FP_DOT_SPEC	**Operation speculatively executed, SME floating-point dot product** The counter counts each speculatively executed floating-point dot product operation counted by SME_FP_SPEC due to an instruction which reads from or writes to any part of the ZA array.
0x8376	SME_FP_MOPA_SPEC	**Operation speculatively executed, SME floating-point outer product and accumulate, or outer product and subtract** The counter counts each speculatively executed floating-point outer product and accumulate, or outer product and subtract operations counted by SME_FP_SPEC due to an instruction which reads from or writes to any part of the ZA array.
0x8378	SME_INT_SPEC	**Operation speculatively executed, SME integer** The counter counts each speculatively executed integer operation counted by SE_INT_SPEC due to an instruction which reads from or writes to any part of the ZA or ZT array.
0x837A	SME_INT_MUL_SPEC	**Operation speculatively executed, SME integer multiply or multiply-accumulate** The counter counts each speculatively executed integer multiply, multiply-add, or multiply-subtract operation counted by SE_INT_MUL_SPEC due to an instruction which reads from or writes to any part of the ZA array.
0x837C	SME_INT_DOT_SPEC	**Operation speculatively executed, SME integer dot product** The counter counts each speculatively executed integer dot product operation counted by SE_INT_DOT_SPEC due to an instruction which reads from or writes to any part of the ZA array.
0x837E	SME_INT_MOPA_SPEC	**Operation speculatively executed, SME integer outer product and accumulate, or outer product and subtract** The counter counts each speculatively executed integer outer product and accumulate, or outer product and subtract operation counted by SME_INT_SPEC due to an instruction which reads from or writes to any part of the ZA array.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x8381	SME_PRED2_NOT_FULL_SPEC	**Operation speculatively executed, SME 2D predicated with at least one inactive element** The counter counts each speculatively executed predicated 2D SME operation which targets the ZA array counted by SME_PRED2_SPEC where at least one element is Inactive. That is, at least one element in the Governing predicates is FALSE. For outer product instructions which are widening, predication is considered with respect to the input element size.
0x8384	SME_PRED2_SPEC	**Operation speculatively executed, SME 2D predicated** The counter counts each speculatively executed 2D operation which targets the ZA array counted by SVE_PRED_SPEC due to an SME instruction with a Governing predicate operand that determines the Active elements.
0x8385	SME_PRED2_EMPTY_SPEC	**Operation speculatively executed, SME 2D predicated with no active element** The counter counts each speculatively executed predicated 2D SME operation which targets the ZA array counted by SME_PRED2_NOT_FULL_SPEC where all elements are Inactive. That is, all elements in the Governing predicates are FALSE. For outer product instructions which are widening, predication is considered with respect to the input element size.
0x8386	SME_PRED2_FULL_SPEC	**Operation speculatively executed, SME 2D predicated with all active elements** The counter counts each speculatively executed predicated 2D SME operation which targets the ZA array counted by SME_PRED2_SPEC where all elements are Active. That is, all elements in the Governing predicates are TRUE. For outer product instructions which are widening, predication is considered with respect to the input element size
0x8387	SME_PRED2_PARTIAL_SPEC	**Operation speculatively executed, SME 2D predicated with partially active elements** The counter counts each speculatively executed predicated 2D SME operation which targets the ZA array counted by SME_PRED2_NOT_FULL_SPEC where neither all elements are Active nor all elements are Inactive. That is, the elements in the Governing predicates are neither all TRUE nor all FALSE. For outer product instructions which are widening, predication is considered with respect to the input element size.
0x8388	SME_LDST_ZAREG_SPEC	**SME ZA unpredicated load/store** The counter counts each speculatively executed operation that reads from or writes to memory counted by SME_LDST_REG_SPEC that was due to an unpredicated instruction targeting the ZA array
0x8389	SME_LD_ZAREG_SPEC	**SME ZA unpredicated load.** The counter counts each speculatively executed operation that reads from memory counted by SME_LDST_ZAREG_SPEC that was due to an unpredicated instruction targeting the ZA array

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x838A	SME_ST_ZAREG_SPEC	**SME ZA unpredicated store** The counter counts each speculatively executed operation that writes to memory counted by SME_LDST_ZAREG_SPEC that was due to an unpredicated instruction targeting the ZA array.
0x838C	SME_LDST_ZTREG_SPEC	**SME ZT unpredicated load/store** The counter counts each speculatively executed operation that reads from or writes to memory counted by SME_LDST_REG_SPEC that was due to an unpredicated instruction targeting the ZT register
0x838D	SME_LD_ZTREG_SPEC	**SME ZT unpredicated load** The counter counts each speculatively executed operation that reads from memory counted by SME_LDST_ZTREG_SPEC that was due to an unpredicated instruction targeting the ZT register
0x838E	SME_ST_ZTREG_SPEC	**SME ZT unpredicated store** The counter counts each speculatively executed operation that writes to memory counted by SME_LDST_ZTREG_SPEC that was due to an unpredicated instruction targeting the ZT register.
0x8390	SME_LDST_REG_SPEC	**SME unpredicated load/store** The counter counts each speculatively executed operation that reads from or writes to memory counted by SME_SPEC that was due to an unpredicated instruction targeting the ZA array or ZT register.
0x8391	SME_LD_REG_SPEC	**SME unpredicated load** The counter counts each speculatively executed operation that reads from memory counted by SME_LDST_REG_SPEC that was due to an unpredicated instruction targeting the ZA array or ZT register.
0x8392	SME_ST_REG_SPEC	**SME unpredicated store** The counter counts each speculatively executed operation that writes to memory counted by SME_LDST_REG_SPEC that was due to an unpredicated instruction targeting the ZA array or ZT register
0x8394	SME_LDST_TILE_SPEC	**SME predicated tile load/store** The counter counts each speculatively executed operation that reads from or writes to memory counted by SME_LDST_REG_SPEC that was due to an instruction with at least one governing predicate which is targeting the ZA array.
0x8395	SME_LD_TILE_SPEC	**SME predicated tile load** The counter counts each speculatively executed operation that reads from memory counted by SME_LDST_TILE_SPEC that was due to an instruction with at least one governing predicate which is targeting the ZA array.
0x8396	SME_ST_TILE_SPEC	**SME predicated tile store** The counter counts each speculatively executed operation that writes to memory counted by SME_LDST_TILE_SPEC that was due to an instruction with at least one governing predicate which is targeting the ZA array.
0x839A	SME_LUT_SPEC	**Lookup table operation speculatively executed, SME** The counter counts each speculatively executed LUT operation counted by SME_SPEC that returns a value from a lookup table made up of a ZT register which is indexed by values from a Z register

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

12.2 Implementation-defined performance monitoring unit events

The C1-SME2 unit Performance Monitoring Unit (PMU) collects events from other units in the design and uses numbers to reference these events.

The C1-SME2 unit PMU collects **IMPLEMENTATION DEFINED** events. The following table shows the **IMPLEMENTATION DEFINED** performance monitors events. These events are not exported to the trace unit. The table also shows the bit position of each event on the event bus. Event numbers that are not listed are reserved.

For more information about these PMU events, see the *Arm® Architecture Reference Manual for A-profile architecture*.

Table 12-2: Implementation defined PMU events

Event number	Mnemonic	Description
`0x3219`	SSVE_INST_SPEC	**Operation speculatively executed, Streaming SVE, including load and store** The counter counts each instruction counted by SVE_INST_SPEC when the CPU executes in Streaming mode.
`0x321A`	SSVE_SPEC	**Operation speculatively executed, Streaming SVE** The counter counts each operation counted by SVE_SPEC specifically in Streaming mode.
`0x321B`	SSVE_LDST_SPEC	**Operation speculatively executed, Streaming SVE load, store, or prefetch** The counter counts each Load or Store operation counted by `SVE_LDST_SPEC` when CPU executes in Streaming mode
`0x321C`	SSVE_LD_SPEC	**Operation speculatively executed, Streaming SVE load** The counter counts each Load operation counted by `SVE_LD_SPEC` when CPU executes in Streaming mode
`0x321D`	SSVE_ST_SPEC	**Operation speculatively executed, Streaming SVE store** The counter counts each Store operation counted by `SVE_ST_SPEC` when CPU executes in Streaming mode
`0x321F`	SSVE_INT_SPEC	**Operation speculatively executed, Streaming SVE integer** The counter counts each Speculatively executed integer arithmetic operation due to an SVE data-processing instruction listed in SVE integer instructions. This counter only counts operations executed in Streaming mode.
`0x3220`	SSVE_FP_SPEC	**Operation speculatively executed, Streaming SVE floating-point** The counter counts each Speculatively executed floating-point operation due to an SVE instruction, specifically in streaming mode.
`0x3221`	SSVE_INT8_SPEC	**Operation speculatively executed, Streaming SVE 8-bit integer** The counter counts each operation counted by SSVE_INT_SPEC where the largest type is 8-bit integer.
`0x3222`	SSVE_INT16_SPEC	**Operation speculatively executed, Streaming SVE 16-bit integer** The counter counts each operation counted by SSVE_INT_SPEC where the largest type is 16-bit integer.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x3223	SSVE_INT32_SPEC	**Operation speculatively executed, Streaming SVE 32-bit integer** The counter counts each operation counted by SSVE_INT_SPEC where the largest type is 32-bit integer.
0x3224	SSVE_INT64_SPEC	**Operation speculatively executed, Streaming SVE 64-bit integer** The counter counts each operation counted by SSVE_INT_SPEC where the largest type is 64-bit integer.
0x3225	SSVE_FP_HP_SPEC	**Operation speculatively executed, Streaming SVE half-precision floating-point** The counter counts each Speculatively executed half-precision floating-point operation due to an SVE instruction, specifically in streaming mode.
0x3226	SSVE_FP_BF16_SPEC	**Operation speculatively executed, Streaming SVE BFloat16 floating-point** The counter counts each Speculatively executed BFloat16 floating-point operation due to an SVE instruction, specifically in streaming mode.
0x3227	SSVE_FP_SP_SPEC	**Operation speculatively executed, Streaming SVE single-precision floating-point** The counter counts each Speculatively executed single-precision floating-point operation due to an SVE instruction, specifically in streaming mode.
0x3228	SSVE_FP_DP_SPEC	**Operation speculatively executed, Streaming SVE double-precision floating-point** The counter counts each Speculatively executed double-precision floating-point operation due to an SVE instruction, specifically in streaming mode.
0x3229	SSVE_INT_MUL_SPEC	**Operation speculatively executed, Streaming SVE integer multiply or multiply-accumulate** The counter counts each Speculatively executed integer multiply or multiply-accumulate operation counted by SSVE_INT_SPEC, specifically in Streaming mode, due to any of the following instructions: • SVE: MAD, MLA, MLS, MSB, MUL, SMULH, or UMULH. • SVE2: CMLA, MLA, MLS, MUL, PMUL, SMLALB, SMLALT, SMLSLB, SMLSLT, SMULH, SMULLB, SMULLT, SQDMLALB, SQDMLALBT, SQDMLALT, SQDMLSLB, SQDMLSLBT, SQDMLSLT, SQDMULH, SQDMULLB, SQDMULLT, SQRDCMLAH, SQRDMLAH, SQRDMLSH, SQRDMULH, UMLALB, UMLALT, UMLSLB, UMLSLT, UMULH, UMULLB, or UMULLT.
0x322A	SSVE_INT_DOT_SPEC	**Operation speculatively executed, Streaming SVE integer dot product** The counter counts each integer dot product operation counted by ASE_SVE_INT_DOT_SPEC due to any of the following instructions: • SVE: SDOT, SUDOT, UDOT, or USDOT. • SVE2: CDOT, SUDOT, or USDOT.
0x322B	SSVE_FP_ADDSUB_SPEC	**Operation speculatively executed, Streaming SVE floating-point add or subtract** The counter counts each Speculatively executed floating-point add or subtract operation counted by SSVE_FP_SPEC due to any of the following instructions: • SVE: FABD, FADD, FSUB, or FSUBR.
0x322C	SSVE_FP_MUL_SPEC	**Operation speculatively executed, Streaming SVE floating-point multiply** The counter counts each Speculatively executed floating-point multiply operation counted by SSVE_FP_SPEC due to any of the following instructions: • SVE: FMUL, FMULX, or FTSMUL.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x322D	SSVE_FP_FMA_SPEC	**Operation speculatively executed, Streaming SVE floating-point multiply-add or multiply-subtract** The counter counts each Speculatively executed floating-point fused multiply-add or multiply-subtract operation counted by FP_FMA_SPEC, specifically in Streaming mode, due to any of the following instructions: • SVE: BFMLALB (vectors), BFMLALT (vectors), FCMLA (vectors), FMAD, FMLA (vectors), FMLS (vectors), FMSB, FNMAD, FNMLA, FNMLS, FNMSB, or FTMAD. • SVE2: BFMLALB (vectors), BFMLALT (vectors), FMLALB (vectors), FMLALT (vectors), FMLSLB (vectors), or FMLSLT (vectors).
0x322E	SSVE_FP_DOT_SPEC	**Operation speculatively executed, Streaming SVE floating-point dot product** The counter counts each dot-product operation counted by SSVE_FP_SPEC due to any of the following instructions: • SVE: BFDOT. • SVE2: BFDOT.
0x322F	SSVE_FP_SQRT_SPEC	**Floating-point operation speculatively executed, Streaming SVE square root** The counter counts each Speculatively executed floating-point square-root operation counted by SSVE_FP_SPEC, specifically in Streaming mode, due to any of the following instructions: • SVE: FSQRT.
0x3230	SSVE_FP_DIV_SPEC	**Floating-point operation speculatively executed, Streaming SVE divide** The counter counts each Speculatively executed floating-point divide operation counted by FP_DIV_SPEC, specifically in Streaming mode, due to any of the following instructions: • SVE: FDIV or FDIVR.
0x3231	SSVE_FP_RECPE_SPEC	**Floating-point operation speculatively executed, Streaming SVE reciprocal estimate** The counter counts each Speculatively executed floating-point reciprocal estimate operation counted by SSVE_FP_SPEC, specifically in Streaming mode, due to any of the following instructions: • SVE: FRECPE or FRSQRTE.
0x3232	SSVE_FP_CVT_SPEC	**Floating-point operation speculatively executed, Streaming SVE convert** The counter counts each Speculatively executed floating-point convert operation counted by SSVE_FP_SPEC, specifically in Streaming mode, due to an SVE instruction.
0x3233	SSVE_FP_VREDUCE_SPEC	**Floating-point operation speculatively executed, Streaming SVE vector reduction** The counter counts each Speculatively executed floating-point treewise reduction operation counted by SSVE_FP_SPEC, specifically in Streaming mode, due to any of the following A64 instructions: • SVE: FADDV, FMAXNMV, FMAXV, FMINNMV, or FMINV.
0x3234	SSVE_PRED_SPEC	**Operation speculatively executed, Streaming SVE predicated** Counts operations counted by SVE_PRED_SPEC, but in Streaming mode only. Note: this counts SME operations requiring PSTATE.ZA to be set, including 2D operations.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x3235	SSVE_PRED_EMPTY_SPEC	**Operation speculatively executed, Streaming SVE predicated with no active predicates** Counts operations counted by SVE_PRED_EMPTY_SPEC, but in Streaming mode only.
0x3236	SSVE_PRED_FULL_SPEC	**Operation speculatively executed, Streaming SVE predicated with all active predicates** Counts speculatively executed predicated SVE operations with all predicate elements active, specifically in Streaming mode.
0x3237	SSVE_PRED_NOT_FULL_SPEC	**Operation speculatively executed, Streaming SVE predicated with no active or partially active predicates** Counts speculatively executed predicated SVE operations with at least one non active predicate elements, specifically in Streaming mode.
0x3238	SSVE_PRED_PARTIAL_SPEC	**Operation speculatively executed, Streaming SVE predicated with partially active predicates** Counts speculatively executed predicated SVE operations with at least one but not all active predicate elements, specifically in streaming mode.
0x3239	SSVE_FP_SCALE_OPS_SPEC	**Scalable floating-point element ALU operations speculatively executed in Streaming SVE** The counter counts each Streaming SVE floating-point ALU operation counted by SSVE_FP_SPEC that was speculatively executed. The counter increments by the number of numerical operations carried out by the instruction per 128/element_size of the result.
0x323A	SSVE_FP_HP_SCALE_OPS_SPEC	**Scalable half-precision floating-point element ALU operations in Streaming mode** The counter counts each Streaming SVE floating-point half-precision ALU operation counted by `SSVE_FP_SPEC` that was speculatively executed. The counter increments by the number of numerical operations carried out by the instruction per 128/element_size of the result.
0x323B	SSVE_FP_BF16_SCALE_OPS_SPEC	**Scalable BFloat16 floating-point element ALU operations in Streaming SVE** The counter counts each Streaming SVE floating-point BFloat16 ALU operation counted by `SSVE_FP_SPEC` that was speculatively executed. The counter increments by the number of numerical operations carried out by the instruction per 128/element_size of the result.
0x323C	SSVE_FP_SP_SCALE_OPS_SPEC	**Scalable single-precision floating-point element ALU operations in Streaming SVE** The counter counts each Streaming SVE floating-point single-precision ALU operation counted by `SSVE_FP_SPEC` that was speculatively executed. The counter increments by the number of numerical operations carried out by the instruction per 128/element_size of the result.
0x323D	SSVE_FP_DP_SCALE_OPS_SPEC	**Scalable double-precision floating-point element ALU operations in Streaming SVE** The counter counts each Streaming SVE floating-point double-precision ALU operation counted by `SSVE_FP_SPEC` that was speculatively executed. The counter increments by the number of numerical operations carried out by the instruction per 128/element_size of the result.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
`0x323E`	SSVE_INT_SCALE_OPS_SPEC	**Scalable integer element ALU operations Speculatively executed in Streaming SVE** The counter counts each Streaming SVE integer ALU operation counted by `SSVE_INT_SPEC` that was speculatively executed. The counter increments by the number of numerical operations carried out by the instruction per 128/element_size of the result.
`0x323F`	SSVE_LDST_SCALE_OPS_SPEC	**Scalable load or store element Operations speculatively executed in Streaming SVE** The counter counts each speculatively executed Memory-read operation or Memory-write operation due to either: • An SVE predicated vector load or store instruction other than a replicating LD1R or LD1RQ instruction. • An SME vector load or store instruction The counter increments by the number of elements accessed by the instruction per `128/element_size` vector of the result.
`0x3240`	SSVE_LD_SCALE_OPS_SPEC	**Scalable load element Operations speculatively executed in Streaming SVE** The counter counts each speculatively executed Memory-read operation due to either: • An SVE predicated vector load instruction other than a replicating LD1R or LD1RQ instruction. • An SME vector load instruction The counter increments by the number of elements accessed by the instruction per 128/element_size vector of the result.
`0x3241`	SSVE_ST_SCALE_OPS_SPEC	**Scalable store element Operations speculatively executed in Streaming SVE** The counter counts each speculatively executed Memory-store operation due to either: • An SVE predicated vector store instruction. • An SME vector store instruction The counter increments by the number of elements accessed by the instruction per 128/element_size vector of the result.
`0x3242`	SSVE_LDST_SCALE_BYTES_SPEC	**Scalable load and store bytes Speculatively executed in Streaming SVE** The counter counts each speculatively executed Memory-read operation or Memory-write operation due to either: • An SVE predicated vector load or store instruction other than a replicating LD1R or LD1RQ instruction. • An SME vector load or store instruction For each instruction, the counter is incremented by `(16 / (CSIZE / MSIZE))`, multiplied by the number of transferred vector registers.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
`0x3243`	SSVE_LD_SCALE_BYTES_SPEC	**Scalable load bytes Speculatively executed in Streaming SVE** The counter counts each speculatively executed Memory-read operation due to either: • An SVE predicated vector load instruction other than a replicating LD1R or LD1RQ instruction. • An SME vector load instruction For each instruction, the counter is incremented by `(16 / (CSIZE / MSIZE))`, multiplied by the number of transferred vector registers.
`0x3244`	SSVE_ST_SCALE_BYTES_SPEC	**Scalable store bytes Speculatively executed in Streaming SVE** The counter counts each speculatively executed Memory-write operation due to either: • An SVE predicated vector store instruction. • An SME vector store instruction For each instruction, the counter is incremented by `(16 / (CSIZE / MSIZE))`, multiplied by the number of transferred vector registers.
`0x3245`	SSVE_LDST_FIXED_BYTES_SPEC	**Non-scalable load and store bytes Speculatively executed in Streaming SVE** The counter counts each byte speculatively read or written in SVE streaming due to any of: • Any Advanced SIMD or SVE non-vector load or store operation. • An SVE replicating LD1R or LD1RQ instruction. For each instruction, the counter is incremented by the number of bytes transferred per register multiplied by the number of registers transferred multiplied by the number of transfers made per register. For example, the counter counts bytes as follows: • SVE and Advanced SIMD LD1R instructions increment the counter by `(MSIZE / 8)`. • SVELD1RQ instructions increment the counter by 16.
`0x3246`	CME_CYCLES	**SME2 unit clock cycles** Counts SME2 unit clock cycles (not timer cycles). The clock measured by this event is defined as the physical clock driving the SME2 unit.
`0x3247`	CME_INST_RETIRED	**SME Instruction architecturally executed** Counts instructions that have been architecturally executed inside the SME2 unit.
`0x3248`	CME_OP_RETIRED	**SME Micro-operation architecturally executed** Counts micro-operations that are architecturally executed. This is a count of number of micro-operations retired from the commit queue in a single cycle.
`0x3249`	CME_STALL	**No operation sent for execution inside SME2 unit** Counts cycles when no operations are sent in the SME2 unit to the rename from the frontend or from the rename to the backend for any reason (either frontend or backend stall).
`0x324A`	CME_STALL_FRONTEND	**No operation sent for execution due to the SME2 unit frontend** No operation has been issued, because of the frontend. The counter counts on any cycle when no operations are issued while instruction queues are not full.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x324B	CME_STALL_FRONTEND_CPU	**SME2 unit front-end stall due to arbitrated core not sending instructions** The counter counts cycles when CME_STALL_FRONTEND increments, due to not enough instructions being sent by the CPU.
0x324C	CME_STALL_FRONTEND_OTHER_CPU	**SME2 unit front-end stall due to non-arbitrated cores** The counter counts cycles when CME_STALL_FRONTEND increments, due to instructions received by another CPU in the system.
0x324D	CME_STALL_BACKEND	**No operation sent for execution in the SME2 unit due to the backend** Counts cycles whenever the rename unit in the SME2 unit is unable to send any micro-operations to the backend of the pipeline because of backend resource constraints. Backend resource constraints can include issue stage fullness, execution stage fullness, or other internal pipeline resource fullness.
0x324E	CME_STALL_BACKEND_CORE	**SME2 unit backend stall due to the execution units** Counts cycles when CME_STALL_BACKEND is set, due to DP issue queues not accepting instructions.
0x324F	CME_STALL_BACKEND_MEM	**SME2 unit backend stall due to the memory system** Counts cycles when CME_STALL_BACKEND is set, due to Load-Store issue queues not accepting instructions.
0x3250	CME_STALL_BACKEND_PF	**SME2 unit backend stall due to the prefetcher** Counts cycles when CME_STALL_BACKEND is set, due to prefetcher issue queues not accepting instructions.
0x3251	CME_STALL_BACKEND_MEM_CACHE	**SME2 unit memory system stall due to the cache pipelines** Counts cycles when CME_STALL_BACKEND_MEM is set, with the oldest instruction in at least one of the load issue queues waiting for cache arbitration.
0x3252	CME_STALL_BACKEND_MEM_STORE	**SME2 unit memory system stall due to store backpressure** Counts cycles when CME_STALL_BACKEND_MEM is set, with the oldest instruction in at least one of the store issue queues waiting for the merge buffer.
0x3253	CME_LDST_SPEC	**SME Operation speculatively executed, load or store** Counts load and store operations that have been speculatively executed.
0x3254	CME_LD_SPEC	**SME Operation speculatively executed, load** Counts speculatively executed SME load operations including Single Instruction Multiple Data (SIMD) load operations executed in streaming SVE mode.
0x3255	CME_UNALIGNED_LDST_SPEC	**SME Operation speculatively executed, unaligned load or store** Counts unaligned memory operations issued by the CPU. This event counts unaligned accesses (as defined by the actual instruction), even if they are subsequently issued as multiple aligned accesses.
0x3256	CME_LDST_ALIGN_LAT	**SME2 unit access with additional latency from alignment** Counts the number of memory read and write accesses in a cycle that incurred additional latency, due to the alignment of the address and the size of data being accessed, which results in store crossing a single cache line.
0x3257	CME_UNALIGNED_LD_SPEC	**SME Operation speculatively executed, unaligned load** Counts unaligned memory read operations issued by the CPU. This event counts unaligned accesses (as defined by the actual instruction), even if they are subsequently issued as multiple aligned accesses. The event does not count preload operations (PLD, PLI).

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x3258	CME_LD_ALIGN_LAT	**SME load access with additional latency from alignment** Counts the number of memory read accesses in a cycle that incurred additional latency, due to the alignment of the address and size of data being accessed, which results in load crossing a single cache line.
0x3259	CME_ST_SPEC	**SME Operation speculatively executed, store** Counts speculatively executed SME store operations including Single Instruction Multiple Data (SIMD) load operations executed in streaming SVE mode.
0x325A	CME_UNALIGNED_ST_SPEC	**SME Operation speculatively executed, unaligned store** Counts unaligned memory write operations issued by the CPU. This event counts unaligned accesses (as defined by the actual instruction), even if they are subsequently issued as multiple aligned accesses.
0x325B	CME_ST_ALIGN_LAT	**SME2 unit store access with additional latency from alignment** Counts the number of memory write accesses in a cycle that incurred additional latency, due to the alignment of the address and the size of data being accessed, which results in store crossing a single cache line.
0x325C	CME_PRF_SPEC	**SME operation speculatively executed, Prefetch** Counts speculatively executed operations in Streaming SVE mode that prefetch memory, executed by the SME2 unit. Only the SME RPRFM instructions are counted by this event.
0x325D	CME_DISPATCH_STALL_IQ_DP0	**SME2 unit dispatch stalled due to IQ full, DP0** Count cycles counted by CME_STALL_BACKEND when at least one operation should be sent to Issue Queue DP0, and this issue queue is full.
0x325E	CME_DISPATCH_STALL_IQ_DP1	**SME2 unit dispatch stalled due to IQ full, DP1** Count cycles counted by CME_STALL_BACKEND when at least one operation should be sent to Issue Queue DP1, and this issue queue is full.
0x325F	CME_DISPATCH_STALL_IQ_LD	**SME2 unit dispatch stalled due to IQ full, Load queue** Count cycles counted by CME_STALL_BACKEND when at least one operation should be sent to Issue Queue LD, and this issue queue is full.
0x3260	CME_OP_ALU_ISSUE	**SME2 unit ALU operation issued** Count number of operations issued to an ALU execution unit at each cycle.
0x3261	CME_OP_MAC_ISSUE	**SME2 unit Multiply-accumulate operation issued** Count number of operations issued to a MAC execution unit at each cycle.
0x3262	CME_OP_PERM_ISSUE	**SME2 unit Permutation operation issued** Count number of operations issued to a Permute execution unit at each cycle.
0x3263	CME_OP_ST_ISSUE	**SME2 unit Store operation issued** Count number of operations issued to a Store execution unit at each cycle.
0x3264	CME_OP_MMDP_ISSUE	**SME2 unit Matrix Multiplication Data-processing operation issued** Count number of operations issued to a Matmul Datapath execution unit at each cycle.
0x3265	CME_OP_MMMV_ISSUE	**SME2 unit Matrix Multiplication Move operation issued** Count number of operations issued to a Matmul Move execution unit at each cycle.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x3266	CME_BUS_REQ_RD_PERCYC	**SME2 unit Bus read transactions in progress** Counts memory transaction requests issued by the CPU to the external bus, including snoop requests and snoop responses in progress on a processor cycle.
0x3267	CME_BUS_REQ	**SME2 unit Bus request** Counts memory transaction requests issued by the CPU to the external bus, including snoop requests and snoop responses.
0x3268	CME_BUS_REQ_RD	**SME2 unit Bus request, read** Counts memory read transaction requests issued by the CPU to the external bus, including snoop requests and snoop responses.
0x3269	CME_BUS_REQ_WR	**SME2 unit Bus request, write** Counts memory write transaction requests issued by the CPU to the external bus, including snoop requests and snoop responses.
0x326A	CME_BUS_CYCLES	**SME2 unit Bus cycle** Counts bus cycles in the SME2 unit. Bus cycles represent a clock cycle in which a transaction could be sent or received on the interface from the SME2 unit to the external bus.
0x326B	CME_BUS_ACCESS	**SME2 unit Bus access** Counts memory transactions issued by the SME2 unit to the external bus, including snoop requests and snoop responses. Each beat of data is counted individually.
0x326C	CME_BUS_ACCESS_RD	**SME2 unit Bus access, read** Counts memory read transactions seen on the external bus. Each beat of data is counted individually.
0x326D	CME_BUS_ACCESS_WR	**SME2 unit Bus access, write** Counts memory write transactions seen on the external bus. Each beat of data is counted individually.
0x326E	CME_DSNP_HIT	**SME2 unit Snoop hit** This event counts each data snoop that hits in a cache outside of the SME2 unit cache.
0x326F	CME_L1D_CACHE	**SME2 unit Level 1 data cache access** Counts each memory-read operation or memory-write operation that causes a level 1 data cache accesses. Each access to a cache line is counted including the multiple accesses caused by single instructions such as multiple-vector instructions. Each access to other level 1 data or unified memory structures, for example refill buffers, write buffers, and write-back buffers, are also counted.
0x3270	CME_L1D_CACHE_HIT	**Level 1 data cache hit, any access** Counts each access counted by L1D_CACHE that hits in the level 1 data cache.
0x3271	CME_L1D_CACHE_HIT_RW	**SME2 unit Level 1 data cache demand hit** Counts cache line hits in the level 1 data cache due to Load or Store operations
0x3272	CME_L1D_CACHE_HIT_RD	**SME2 unit Level 1 data cache demand hit, read** Counts cache line hits in the level 1 data cache due to Load operations
0x3273	CME_L1D_CACHE_HIT_WR	**SME2 unit Level 1 data cache demand hit, write** Counts cache line hits in the level 1 data cache due to Store operations

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x3274	CME_L1D_CACHE_HIT_RW_FHWPRF	**SME2 unit Level 1 data cache demand access first hit, fetched by hardware prefetcher** The counter counts each demand access first hit counted by L1D_CACHE_HIT_RW where the cache line was fetched by a hardware prefetcher. That is, the L1D_CACHE_REFILL_HWPRF event was generated when the cache line was fetched into the cache. Only the first hit by a demand access is counted. After this event is generated for a cache line, the event is not generated again for the same cache line while it remains in the cache.
0x3275	CME_L1D_CACHE_MISS	**SME2 unit Level 1 data cache demand access miss** Counts access counted by L1D_CACHE that miss in the level 1 data cache.
0x3276	CME_L1D_CACHE_RW	**SME2 unit Level 1 data cache demand access** Counts level 1 data demand cache accesses from any load or store operation.
0x3277	CME_L1D_CACHE_RD	**SME2 unit Level 1 data cache demand access, read** Counts level 1 data cache accesses from any load operation.
0x3278	CME_L1D_CACHE_WR	**SME2 unit Level 1 data cache demand access, write** Counts level 1 data cache accesses generated by store operations.
0x3279	CME_L1D_CACHE_PRF	**SME2 unit Level 1 data cache, preload or prefetch hit** Counts level 1 data cache accesses from hardware prefetcher, software preload or prefetch instructions.
0x327A	CME_L1D_CACHE_HWPRF	**SME2 unit Level 1 data cache hardware prefetch** The counter counts each access counted by L1D_CACHE that is due to a hardware prefetch. The hardware prefetch is generated by a hardware prefetcher at the Level 1 data or unified cache.
0x327B	CME_L1D_CACHE_PRFM	**SME2 unit Level 1 instruction cache software preload** Counts level 1 data cache accesses from software preload or prefetch instructions.
0x327C	CME_L1D_CACHE_REFILL	**SME2 unit Level 1 data cache refill** Counts each access counted by L1D_CACHE that causes a level 1 data cache refill due to a miss in the level 1 data cache. This event only counts one event per cache line.
0x327D	CME_L1D_CACHE_REFILL_INNER	**SME2 unit Level 1 data cache refill, inner** Counts level 1 data cache refills where the cache line data came from caches inside the immediate cluster of the core.
0x327E	CME_L1D_CACHE_REFILL_OUTER	**SME2 unit Level 1 data cache refill, outer** Counts level 1 data cache refills for which the cache line data came from outside the immediate cluster of the core, like an SLC in the system interconnect or DRAM.
0x3280	CME_L1D_CACHE_REFILL_RD	**SME2 unit Level 1 data cache refill, read** Counts level 1 data cache refills caused by speculatively executed load instructions where the memory read operation misses in the level 1 data cache. This event only counts one event per cache line.
0x3281	CME_L1D_CACHE_REFILL_WR	**SME2 unit Level 1 data cache refill, write** Counts level 1 data cache refills caused by speculatively executed store instructions where the memory write operation misses in the level 1 data cache. This event only counts one event per cache line.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x3282	CME_L1D_CACHE_INVAL	**SME2 unit Level 1 data cache invalidate** Counts each explicit invalidation of a cache line in the level 1 data cache caused by broadcast cache coherency operations from another CPU in the system. This event does not count for the following conditions: 1. A cache refill invalidates a cache line. 2. Invalidation during hardware power-off sequence.
0x3283	CME_L1D_CACHE_LMISS_RD	**SME2 unit Level 1 data cache long-latency read miss** Counts cache line refills into the level 1 data cache from any memory read operations, that incurred additional latency.
0x3284	CME_L1D_CACHE_WB	**SME2 unit Level 1 data cache write-back** Counts write-backs of dirty data from the L1 data cache to the next cache. This occurs when either a dirty cache line is evicted from L1 data cache and allocated in the next cache or dirty data is written to the next cache and possibly to the next level of cache. This event counts both victim cache line evictions and cache write-backs from snoops or cache maintenance operations. The following cache operations are not counted: 1. Invalidations which do not result in data being transferred out of the L1 (such as evictions of clean data), 2. Full line writes which write to next level of cache without writing L1, such as write streaming mode.
0x3285	CME_L1D_CACHE_WB_CLEAN	**SME2 unit Level 1 data cache write-back, cleaning and coherency** Counts write-backs from the level 1 data cache that are a result of a coherency operation made by another CPU. Event count includes cache maintenance operations.
0x3286	CME_L1D_CACHE_WB_VICTIM	**SME2 unit Level 1 data cache write-back, victim** Counts dirty cache line evictions from the level 1 data cache caused by a new cache line allocation. This event does not count evictions caused by cache maintenance operations.
0x3287	CME_L3D_CACHE	**Level 3 data cache access due to the SME2 unit** Counts level 3 cache accesses. Level 3 cache is a unified cache for data and instruction accesses. Accesses are for misses in the lower level caches or translation resolutions due to accesses.
0x3288	CME_L3D_CACHE_RW	**Level 3 data cache demand access due to the SME2 unit** Counts level 3 cache accesses caused by any memory read or write operation. level 3 cache is a unified cache for data and instruction accesses. Accesses are for misses in the lower level caches or translation resolutions due to accesses.
0x3289	CME_L3D_CACHE_RD	**Level 3 data cache demand access due to the SME2 unit, read** Sum of L3 cache reads from all Pipelines
0x328A	CME_L3D_CACHE_WR	**Level 3 data cache demand access due to the SME2 unit, write** L3 cache write
0x328B	CME_L3D_CACHE_ALLOCATE	**Level 3 data cache allocation due to the SME2 unit, without refill** Counts level 3 cache line allocates that do not fetch data from outside the level 3 data or unified cache. For example, allocates due to streaming stores.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x328D	CME_L3D_CACHE_LMISS_RD	**Level 3 data cache long-latency read miss due to the SME2 unit** Counts any cache line refill into the level 3 cache from memory read operations that incurred additional latency.
0x328E	CME_L3D_CACHE_HIT	**Level 3 data cache hit due to the SME2 unit** Counts level 3 cache accesses that hit in the level 3 cache.
0x328F	CME_L3D_CACHE_MISS	**Level 3 data cache demand access miss due to the SME2 unit** Counts level 3 cache accesses that missed in the level 3 cache.
0x3290	CME_L3D_CACHE_REFILL	**Level 3 data cache refill due to the SME2 unit** Counts level 3 accesses that receive data from outside the L3 cache.
0x3291	CME_L3D_CACHE_REFILL_PRFM	**Level 3 data cache refill due to the SME2 unit, software preload** Counts cacheable reads generated by hardware or software prefetches that receive data from outside the L3 cache.
0x3292	CME_L3D_CACHE_REFILL_RD	**Level 3 data cache refill due to the SME2 unit, read** • This event duplicates L3D_CACHE_REFILL. • If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.
0x3293	CME_LL_CACHE	**Last level cache access due to the SME2 unit** Counts transactions that were returned from outside the core cluster. This event counts when the system register CMECFG.EXTLLC bit is set.
0x3294	CME_LL_CACHE_RD	**Last level cache access due to the SME2 unit, read** Counts read transactions that were returned from outside the core cluster. This event counts when the system register CMECFG.EXTLLC bit is set. This event counts read transactions returned from outside the core if those transactions are either hit in the system level cache or missed in the SLC and are returned from any other external sources. This event is a superset of the CME_LL_CACHE_MISS_RD event.
0x3295	CME_LL_CACHE_MISS_RD	**Last level cache miss due to the SME2 unit** Counts read transactions that were returned from outside the core cluster but missed in the system level cache. This event counts when the system register CMECFG.EXTLLC bit is set. This event counts read transactions returned from outside the core if those transactions are missed in the System level Cache. The data source of the transaction is indicated by a field in the CHI transaction returning to the CPU. This event does not count reads caused by cache maintenance operations. This event is a subset of the CME_LL_CACHE_RD event.
0x3296	CME_LL_CACHE_HIT	**Last level cache hit due to the SME2 unit** Counts transactions that were returned from outside the core cluster, hitting in the last. level cache. This event counts when the system register CMECFG.EXTLLC bit is set.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x3298	CME_MEM_ACCESS	**SME2 unit Data memory access** Counts memory accesses issued by the SME2 unit, where those accesses are issued due to load or store operations. This event counts memory accesses no matter whether the data is received from any level of cache hierarchy or external memory. If memory accesses are broken up into smaller transactions than what were specified in the load or store instructions, then the event counts those smaller memory transactions.
0x3299	CME_MEM_ACCESS_RD	**SME2 unit Data memory access, read** Counts memory accesses issued by the CPU due to load operations. The event counts any memory load access, no matter whether the data is received from any level of cache hierarchy or external memory. The event also counts atomic load operations. If memory accesses are broken up by the load/store unit into smaller transactions that are issued by the bus interface, then the event counts those smaller transactions.
0x329A	CME_MEM_ACCESS_RD_PERCYC	**Number of outstandanding memory reads per cycle in the SME2 unit** Counts the number of outstanding loads or memory read accesses per cycle.
0x329B	CME_MEM_ACCESS_WR	**SME2 unit Data memory access, write** Counts memory accesses issued by the SME2 unit due to store operations. The event counts any memory store access, no matter whether the data is located in any level of cache or external memory. If memory accesses are broken up by the load/store unit into smaller transactions that are issued by the bus interface, then the event counts those smaller transactions.
0x329C	CME_REMOTE_ACCESS	**Access to another socket in a multi-socket system due to the SME2 unit** Counts accesses to another chip, which is implemented as a different CMN mesh in the system. If the CHI bus response back to the core indicates that the data source is from another chip (mesh), then the counter is updated. If no data is returned, even if the system snoops another chip/mesh, then the counter is not updated.
0x329D	CME_REMOTE_ACCESS_RD	**Access to another socket in a multi-socket system due to the SME2 unit, read** Counts read accesses to another chip, which is implemented as a different CMN mesh in the system. If the CHI bus response back to the core indicates that the data source is from another chip (mesh), then the counter is updated. If no data is returned, even if the system snoops another chip/mesh, then the counter is not updated.
0x32A4	SSVE_SLOW_INSTR	**Slow Streaming SVE instructions** Number of instructions which are considered as slow instructions (communication to the CPU, serializing instruction)
0x32A5	SSVE_GPR_UPDATE	**Slow Streaming SVE instructions producing a General purpose register** Number of instructions producing a GPR register (not XZR/WZR) from SME2 unit to the CPU
0x32A6	SSVE_PRED_UPDATE	**Slow Streaming SVE instructions producing a predicate register** Number of instructions producing a Predicate register from SME2 unit to the CPU
0x32A7	SSVE_FLAG_UPDATE	**Slow Streaming SVE instructions producing condition flags** Number of instructions producing a Flag register (CPSR) from SME2 unit to the CPU

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
0x32A8	SSVE_CONTEXT_SWITCH	**Context switch stall cycles** Cycles inside the SME2 unit lost for context switch (new instructions are stalled in decode stage). Counted for a core by the time arbitration acknowledge is sent to the time first instruction can be executed.
0x32A9	SSVE_FAST_CONTEXT_SWITCH	**Stall cycles due to another Core accessing its context** Cycles inside the SME2 unit lost because a core loads or stores its context (without causing a real context switch).
0x32AA	CME_OP_DIVSQRT_ISSUE	**SME2 unit Divide or Square Root operation issued** Count number of operations issued to a DIV/SQRT execution unit at each cycle.
0x32AB	SSVE_UNPRED_SPEC	**Operation speculatively executed, Streaming SVE unpredicated** The counter counts each Speculatively executed data-processing, load, or store operation due to an Streaming SVE instruction without a Governing predicate operand. This counts the SME operations requiring PSTATE.ZA to be set. It does not count Advanced SIMD operations.
0x32AC	CME_DRAM_ACCESS	**Access to DRAM due to the SME2 unit** Counts access where the data was sourced from the DRAM.
0x32AD	CME_L1D_LFB_HIT_RW_FHWPRF	**Level 1 data cache demand access line-fill buffer first hit, recently fetched by hardware prefetcher** The counter counts each demand access first hit in outstanding where the cache line was fetched by a hardware prefetcher. Only the first hit by a demand access is counted. After this event is generated for a cache line, the event is not generated again for the same cache line while it remains in the cache.
0x32AF	SME_INT_OTHER_SPEC	**Operation speculatively executed, other SME integer** The counter counts each speculatively executed operation counted by SME_INT_SPEC due to an instruction which reads from or writes to any part of the ZA array, which is not counted by SME_INT_DOT_SPEC, SME_INT_MOPA_SPEC or SME_INT_MUL_SPEC
0x32B0	SME_FP_OTHER_SPEC	**Operation speculatively executed, other SME floating-point** The counter counts each speculatively operation counted by SME_FP_SPEC due to an instruction which reads from or writes to any part of the ZA array, which is not counted by SME_FP_ADDSUB_SPEC, SME_FP_DOT_SPEC, SME_FP_FMA_SPEC or SME_FP_MOPA_SPEC
0x32B1	CME_L1D_CACHE_REFILL_PRFM	**SME2 unit Level 1 data cache refill, software preload** Counts level 1 data cache refills where the cache line access was generated by software preload or prefetch instructions.
0x32B2	CME_L1D_CACHE_REFILL_HWPRF	**SME2 unit Level 1 data cache refill, hardware preload** Counts each hardware prefetch counted by L1D_CACHE_HWPRF that causes a refill of the Level 1 data or unified cache from outside of the Level 1 data or unified cache.
0x32B3	CME_L3D_CACHE_PRFM	**Level 3 data cache software preload due to the SME2 unit** Counts level 3 cache accesses generated by software prefetches.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Event number	Mnemonic	Description
`0x32B4`	CME_L3D_CACHE_HWPRF	**SME2 unit Level 3 data cache hardware prefetch** Counts level 3 cache accesses generated by hardware prefetches.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

13. Activity Monitors Extension support

The C1-SME2 unit 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 configurable from the utility bus only.

The C1-SME2 unit 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.

13.1 Activity monitors access

The C1-SME2 unit supports memory-mapped access using the utility bus interface.

For information on the memory mapping of these registers, see the *Arm® Architecture Reference Manual for A-profile architecture*.

External memory-mapped access

Activity monitors are memory-mapped accessed from the utility bus interface.

The base address for Activity Monitoring Unit (AMU) registers on the utility bus interface is `0x<m+NUM_CORES>90000`, where:

- m is the C1-SME2 unit instance number in the C1-DSU cluster. m has a value of either 0 or 1
- NUM_CORES is the number of cores in the cluster

For example:

- For SME2 0, AMU register base address is `0x<NUM_CORES>90000`
- For SME2 1, AMU register base address is `0x<1+NUM_CORES>90000`

The AMU registers are:

- Read-write in Secure state
- Read-only in Non-secure state

The AMU registers are treated as RAZ/WI if they are marked as Reserved.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

13.2 Activity monitors counters

The C1-SME2 unit implements four activity monitors counters, 0-3, and three auxiliary counters, 10-12 that map to specific Activity Monitoring Unit (AMU) events.

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 can affect any counter. For example, when the C1-SME2 unit is idle and stops the clock or turns off power.
- Events 0-3 and auxiliary events 10-12 are fixed, and the AMEVTYPER0<n> and AMEVTYPER1<n> evtCount bits are read-only.
- The activity monitor counters are reset to zero on a Cold reset of the power domain of the unit. When the unit is not in reset, activity monitoring is available.

13.3 Activity monitors events

Activity monitors events in the C1-SME2 unit 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 13-1: Mapping of counters to fixed events

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR00	CME_CYCLES	`0x3246`	C1-SME2 unit frequency cycles
AMEVCNTR01	CNT_CYCLES	`0x4004`	Constant frequency cycles
AMEVCNTR02	CME_INSTR_RETIRED	`0x3247`	Instruction architecturally executed on the C1-SME2 unit This counter increments for every instruction that is executed architecturally inside the C1-SME2 unit, from any core, including instructions that fail their condition code check.
AMEVCNTR03	CME_STALL_BACKEND_MEM	`0x324F`	Memory stall cycles in the C1-SME2 unit This counter counts cycles in which the C1-SME2 unit is unable to dispatch instructions from the frontend to the backend due to a backend stall caused from any core by a miss in the last level of cache within the C1-SME2 unit 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
AMEVCNTR12	MPMM_THRESHOLD_GEAR2	`0x0302`	Maximum Power Mitigation System (MPMM) Gear 2 activity period threshold exceeded

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR13	CPU_ACTIVITY	0x0310	CPU activity count This counter is an accumulation of CPU activity. This value is updated on a periodic basis with a count of the activity within the CPU. For example, the difference between two reads of the count divided by the time between the counts will be an average activity level for that period.

Related information

4.5.1 Maximum Power Mitigation Mechanism on page 39

13.4 External AMU registers

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped AMU registers in the core.

Table 13-2: AMU registers summary

Offset	Name	Reset	Width	Description
0x0	AMEVCNTR00 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
0x8	AMEVCNTR01 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
0x10	AMEVCNTR02 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
0x18	AMEVCNTR03 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
0x100	AMEVCNTR10 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
0x108	AMEVCNTR11 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
0x110	AMEVCNTR12 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
0x118	AMEVCNTR13 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
0x400	AMEVTYPER00	0x00003246	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	0x00004004	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	0x00003247	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	0x0000324F	32-bit	Activity Monitors Event Type Registers 0
0x480	AMEVTYPER10	0x00000300	32-bit	Activity Monitors Event Type Registers 1
0x484	AMEVTYPER11	0x00000301	32-bit	Activity Monitors Event Type Registers 1
0x488	AMEVTYPER12	0x00000302	32-bit	Activity Monitors Event Type Registers 1
0x48C	AMEVTYPER13	0x00000310	32-bit	Activity Monitors Event Type Registers 1
0xC00	AMCNTENSET0	See individual bit resets.	32-bit	Activity Monitors Count Enable Set Register 0
0xC04	AMCNTENSET1	See individual bit resets.	32-bit	Activity Monitors Count Enable Set Register 1
0xC20	AMCNTENCLR0	See individual bit resets.	32-bit	Activity Monitors Count Enable Clear Register 0
0xC24	AMCNTENCLR1	See individual bit resets.	32-bit	Activity Monitors Count Enable Clear Register 1
0xCE0	AMCGCR	0x00000404	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	0x11003F07	32-bit	Activity Monitors Configuration Register

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Offset	Name	Reset	Width	Description
0xE04	AMCR	See individual bit resets.	32-bit	Activity Monitors Control Register
0xE08	AMIIDR	`0xD8D1243B`	32-bit	Activity Monitors Implementation Identification Register
0xFA8	AMDEVAFF0	`0x00000000`	32-bit	Activity Monitors Device Affinity Register 0
0xFAC	AMDEVAFF1	`0x00000000`	32-bit	Activity Monitors Device Affinity Register 1
0xFBC	AMDEVARCH	`0x47700A66`	32-bit	Activity Monitors Device Architecture Register
0xFCC	AMDEVTYPE	`0x00000016`	32-bit	Activity Monitors Device Type Register
0xFD0	AMPIDR4	`0x00000004`	32-bit	Activity Monitors Peripheral Identification Register 4
0xFE0	AMPIDR0	`0x0000008D`	32-bit	Activity Monitors Peripheral Identification Register 0
0xFE4	AMPIDR1	`0x000000BD`	32-bit	Activity Monitors Peripheral Identification Register 1
0xFE8	AMPIDR2	`0x0000001B`	32-bit	Activity Monitors Peripheral Identification Register 2
0xFEC	AMPIDR3	`0x00000020`	32-bit	Activity Monitors Peripheral Identification Register 3
0xFF0	AMCIDR0	`0x0000000D`	32-bit	Activity Monitors Component Identification Register 0
0xFF4	AMCIDR1	`0x00000090`	32-bit	Activity Monitors Component Identification Register 1
0xFF8	AMCIDR2	`0x00000005`	32-bit	Activity Monitors Component Identification Register 2
0xFFC	AMCIDR3	`0x000000B1`	32-bit	Activity Monitors Component Identification Register 3

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

14. Statistical Profiling Extension support

The C1-SME2 unit implements the Statistical Profiling Extension (SPE) to the Arm®v8.8-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 C1-SME2 unit only reports events to the core that is executing instructions on the C1-SME2 unit. The core is responsible for creating and writing the SPE packets. For more information, see *Statistical Profiling Extension support* in your core Technical Reference Manual (TRM).

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Appendix A AArch64 registers

This appendix contains the descriptions for the C1-SME2 AArch64 registers.

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

A.1 AArch64 Generic System Control registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** Generic System Control registers in the core.

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the *Arm® Architecture Reference Manual for A-profile architecture*.

Table A-1: Generic System Control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CMECFR_EL1	3	0	C15	C11	0	See individual bit resets.	64-bit	SME2 Features Register
IMP_CMESELR_EL1	3	0	C15	C11	1	See individual bit resets.	64-bit	SME2 Selection Register
IMP_CMEACTLR_EL1	3	0	C15	C11	2	See individual bit resets.	64-bit	SME2 Auxiliary Control Register
IMP_CMEACTLR2_EL1	3	0	C15	C11	3	See individual bit resets.	64-bit	SME2 Auxiliary Control Register 2
IMP_CMECFG_EL1	3	0	C15	C11	4	See individual bit resets.	64-bit	SME2 Configuration Register
IMP_CMEPWR_EL1	3	0	C15	C11	5	See individual bit resets.	64-bit	SME2 Power Register
IMP_CMEREVIDR_EL1	3	0	C15	C11	6	`0x0000000000000000`	64-bit	SME2 unit ECO ID Register
IMP_CABACTLR_EL1	3	0	C15	C12	5	See individual bit resets.	64-bit	CAB Auxiliary Control Register
IMP_CABECTLR_EL1	3	0	C15	C12	6	See individual bit resets.	64-bit	CAB Control Register
IMP_CMERAMDATA0_EL3	3	6	C15	C12	2	See individual bit resets.	64-bit	Data0 for RAMINDEX
IMP_CMERAMDATA1_EL3	3	6	C15	C12	3	See individual bit resets.	64-bit	Data1 for RAMINDEX
IMP_CMERAMDATA2_EL3	3	6	C15	C12	4	See individual bit resets.	64-bit	Data2 for RAMINDEX

A.1.1 IMP_CMECFR_EL1, SME2 Features Register

SME2 Features Register

Configurations

This register is available in all configurations.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

64

Functional group

Generic System Control

Access type

RO

Reset value

```
x000 0000 0000 0000 0000 0000 0000 xxxx 0000 0000 0000 0000 0000 0000 0000 xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-1: AARCH64_IMP_CMECFR_EL1 bit assignments

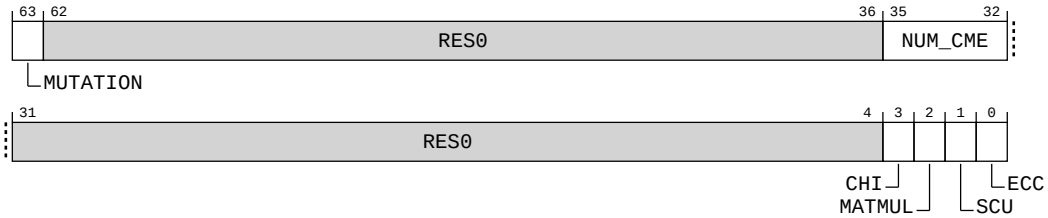


Table A-2: IMP_CMECFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	MUTATION	Indicate support for mutations **0b0** Mutations are not present.	x
[62:36]	**RES0**	Reserved	**RES0**
[35:32]	NUM_CME	Number of SME2 units in the cluster, minus 1.	xxxx
[31:4]	**RES0**	Reserved	**RES0**
[3]	CHI	Number of CHI ports **0b0** One CHI port. **0b1** Two CHI ports.	x

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[2]	MATMUL	MATMUL Selection **0b0** Half matmul. **0b1** Full matmul.	x
[1]	SCU	Indicate support for SCU **0b1** SCU mode.	x
[0]	ECC	ECC selection **0b0** No ECC. **0b1** ECC.	x

Access

`MRS <Xt>, S3_0_C15_C11_0`

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b000

Accessibility

`MRS <Xt>, S3_0_C15_C11_0`

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CMECFR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CMECFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMECFR_EL1;
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

A.1.2 IMP_CMESELR_EL1, SME2 Selection Register

Allows selection of the SME2 units for the CPU to program to SME2 Implementation Defined registers. This register provides information on the maximum index of supported SME2 units in the cluster.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-2: AARCH64_IMP_CMESELR_EL1 bit assignments

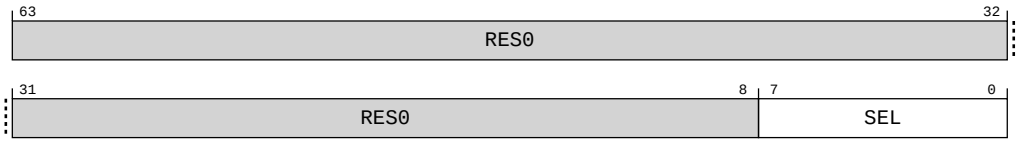


Table A-4: IMP_CMESELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[7:0]	SEL	Select the destination SME2 unit for the CPU If IMP_CMESELR_EL1.SEL is greater than IMP_CMECFR_EL1.NUM_CME, then all of the following apply: • The value read back from IMP_CMESELR_EL1.SEL is unknown. • One of the following occurs: ◦ An **UNKNOWN** SME2 unit is selected. ◦ The IMP_CME*_EL1 registers are **RAZ/WI**. ◦ IMP_CME*_EL1 register reads and writes are NOPs. ◦ IMP_CME*_EL1 register reads and writes are **UNDEFINED**.	8{x}

Access

```
MSR S3_0_C15_C11_1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b001

```
MRS <Xt>, S3_0_C15_C11_1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b001

Accessibility

```
MSR S3_0_C15_C11_1, <Xt>
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.CMEEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.CMEEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMESELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
        IMP_CMESELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CMESELR_EL1 = X[t, 64];
```

```
MRS <Xt>, S3_0_C15_C11_1
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CMESELR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CMESELR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMESELR_EL1;
```

A.1.3 IMP_CMEACTLR_EL1, SME2 Auxiliary Control Register

This register contains control bits that affect the SME2 unit behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure A-3: AARCH64_IMP_CMEACTLR_EL1 bit assignments

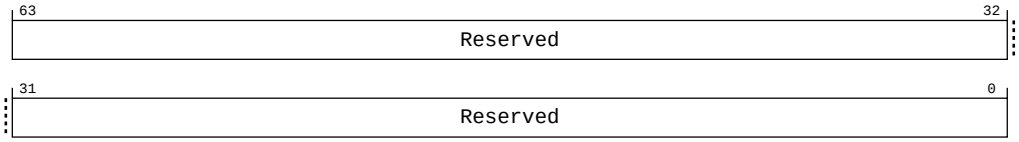


Table A-7: IMP_CMEACTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use	64{x}

Access

```
MSR S3_0_C15_C11_2, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b010

```
MRS <Xt>, S3_0_C15_C11_2
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b010

Accessibility

```
MSR S3_0_C15_C11_2, <Xt>
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.CMEEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.CMEEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMEACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMEACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CMEACTLR_EL1 = X[t, 64];
```

```
MRS <Xt>, S3_0_C15_C11_2
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CMEACTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CMEACTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMEACTLR_EL1;
```

A.1.4 IMP_CMEACTLR2_EL1, SME2 Auxiliary Control Register 2

This register contains control bits that affect the SME2 unit behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-4: AARCH64_IMP_CMEACTLR2_EL1 bit assignments

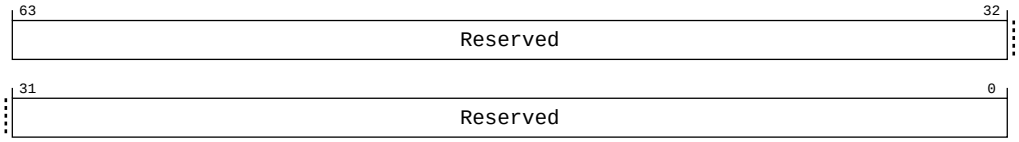


Table A-10: IMP_CMEACTLR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use	`64{x}`

Access

```
MSR S3_0_C15_C11_3, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b011

```
MRS <Xt>, S3_0_C15_C11_3
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b011

Accessibility

```
MSR S3_0_C15_C11_3, <Xt>
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.CMEEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.CMEEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMEACTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMEACTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CMEACTLR2_EL1 = X[t, 64];
```

```
MRS <Xt>, S3_0_C15_C11_3
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CMEACTLR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CMEACTLR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMEACTLR2_EL1;
```

A.1.5 IMP_CMECFG_EL1, SME2 Configuration Register

This register contains control bit that affect the unit behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx x010 1010 1001 1111 0001 0001 0001 0001 0101
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-5: AARCH64_IMP_CMECFG_EL1 bit assignments

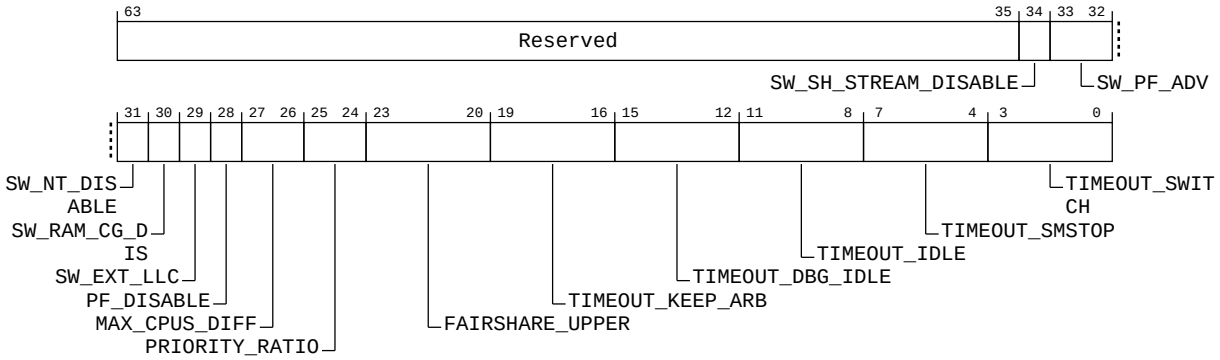


Table A-13: IMP_CMECFG_EL1 bit descriptions

Bits	Name	Description	Reset
[63:35]	Reserved	Reserved	`29{x}`
[34]	SW_SH_STREAM_DISABLE	Streaming of full cache line stores hitting in Shared is disabled **0b0** Streaming of full cache line stores hitting in Shared is enabled. **0b1** Streaming of full cache line stores hitting in Shared is disabled.	`0b0`
[33:32]	SW_PF_ADV	PF global advance budget **0b00** Main memory latency is 90 cycles. **0b01** Main memory latency is 150 cycles. **0b10** Main memory latency is 200 cycles. **0b11** Main memory latency is 250 cycles.	`0b10`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[31]	SW_NT_DISABLE	Non Temporal hint is disabled as well as associated allocation policies **0b0** Non Temporal hint is enabled. **0b1** Non Temporal hint is disabled.	0b1
[30]	SW_RAM_CG_DIS	RAM global clockgate disable **0b0** RAMs clkgate is not disabled. **0b1** RAMs clkgate is disabled.	0b0
[29]	SW_EXT_LLC	External last level cache **0b0** Last Level Cache is not external. **0b1** Last Level Cache is external.	0b1
[28]	PF_DISABLE	Disable the prefetcher	0b0
[27:26]	MAX_CPUS_DIFF	When two SME2 units are implemented (IMP_CMECFR_EL1.NUM_CME == 1), reassign_hint is sent when the difference of assigned CPUs between SME2 units is equal or greater than MAX_CPUS_DIFF. When one SME2 unit is implemented (IMP_CMECFR_EL1.NUM_CME == 0) this field is ignored and reassign_hint is not sent to CPUs.	0b10
[25:24]	PRIORITY_RATIO	Defines a priority ratio applied to each CPU priority level. This is only applicable for CPU using fairshare arbitration scheme. **0b00** No priority ratio applied. **0b01** Apply a ratio of 1/4. **0b10** Apply a ratio of 1/2. **0b11** Apply a ratio of 1. This modifies the number of cycles like 512 x (1 + CPU priority level x PRIORITY_RATIO) x 2^(TIMEOUT_SWITCH)	0b01
[23:20]	FAIRSHARE_UPPER	Defines the higher fairshare priority, priorities above this value are exclusive. • if cpu priority level > FAIRSHARE_UPPER use exclusive arbitration • if cpu priority level <= FAIRSHARE_UPPER use fairshare arbitration	0b1111
[19:16]	TIMEOUT_KEEP_ARB	512x2^(TIMEOUT_KEEP_ARB) is the min number of cycles a cpu can keep the arbitration. Allows a cpu to keep arbitration for a minimum of cycles before being preempted by a cpu with higher priority	0b0001
[15:12]	TIMEOUT_DBG_IDLE	512x2^(TIMEOUT_DBG_IDLE) is the number of cycles of inactivity when arbitrated Core is in Debug State before switching to another Core. This has no impact if the Core in Debug State is using an exclusive priority until SM=0.	0b0001

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[11:8]	TIMEOUT_IDLE	512x2^(TIMEOUT_IDLE) is the number of cycles of inactivity before switching to another cpu. This has no effect for Cores using exclusive priorities until SM=0.	0b0001
[7:4]	TIMEOUT_SMSTOP	512x2^(TIMEOUT_SMSTOP) is the number of cycles before switching to another cpu if arbitrated one has executed SMSTOP (SM=0).	0b0001
[3:0]	TIMEOUT_SWITCH	512x2^(TIMEOUT_SWITCH) is the number of cycles for switching from a arbitrated CPU to another one with same priority. For example: TIMEOUT_SWITCH = 5 corresponds to 512x(2^5)= 16384 cycles	0b0101

Access

```
MSR S3_0_C15_C11_4, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b100

```
MRS <Xt>, S3_0_C15_C11_4
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b100

Accessibility

```
MSR S3_0_C15_C11_4, <Xt>
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.CMEEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.CMEEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMECFG_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMECFG_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
    IMP_CMECFG_EL1 = X[t, 64];
```

```
MRS <Xt>, S3_0_C15_C11_4
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CMECFG_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CMECFG_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMECFG_EL1;
```

A.1.6 IMP_CMEPWR_EL1, SME2 Power Register

This register allows configuration of power behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx 0000 0100 0000 0000 xxxx xxxx xxxx xxxx 0000 0001 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure A-6: AARCH64_IMP_CMEPWR_EL1 bit assignments

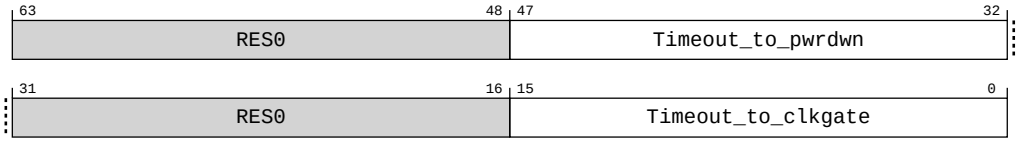


Table A-16: IMP_CMEPWR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:32]	Timeout_to_pwrdwn	Number of system clock cycles counted by virtual counter CNTVCT_EL0 before starting the transition to power-off state	0x0400
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	Timeout_to_clkgate	Number of SME2 unit clock cycles with no activity before SME2 unit clock is gated	0x0100

Access

```
MSR S3_0_C15_C11_5, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b101

```
MRS <Xt>, S3_0_C15_C11_5
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b101

Accessibility

```
MSR S3_0_C15_C11_5, <Xt>
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.CMEEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.CMEEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMEPWR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMEPWR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CMEPWR_EL1 = X[t, 64];
```

```
MRS <Xt>, S3_0_C15_C11_5
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CMEPWR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CMEPWR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMEPWR_EL1;
```

A.1.7 IMP_CMEREVIDR_EL1, SME2 unit ECO ID Register

This register contains the SME2 unit ECO ID

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure A-7: AARCH64_IMP_CMEREVIDR_EL1 bit assignments

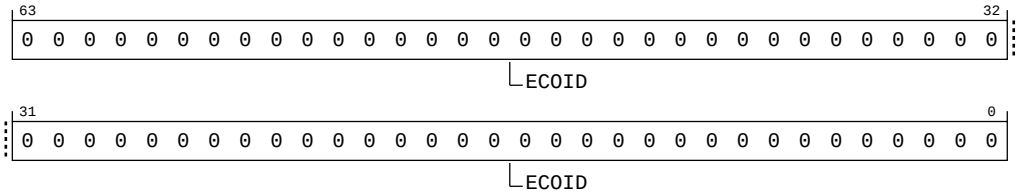


Table A-19: IMP_CMEREVIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	ECOID	**0x0000000000000000** ECOID value for r0p0-00lac0	0x0000000000000000

Access

```
MRS <Xt>, S3_0_C15_C11_6
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1011	0b110

Accessibility

```
MRS <Xt>, S3_0_C15_C11_6
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CMEREVIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CMEREVIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMEREVIDR_EL1;
```

A.1.8 IMP_CABACTLR_EL1, CAB Auxiliary Control Register

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-8: AARCH64_IMP_CABACTLR_EL1 bit assignments

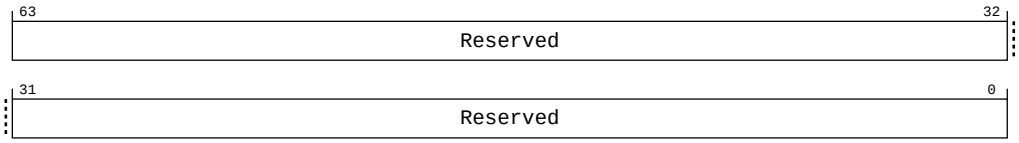


Table A-21: IMP_CABACTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use	`64{x}`

Access

```
MSR S3_0_C15_C12_5, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1100	0b101

```
MRS <Xt>, S3_0_C15_C12_5
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1100	0b101

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

MSR S3_0_C15_C12_5, <Xt>

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.CMEEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.CMEEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CABACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CABACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CABACTLR_EL1 = X[t, 64];
```

MRS <Xt>, S3_0_C15_C12_5

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CABACTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CABACTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CABACTLR_EL1;
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

A.1.9 IMP_CABECTLR_EL1, CAB Control Register

This register contains control bit that affect the CAB unit behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-9: AARCH64_IMP_CABECTLR_EL1 bit assignments

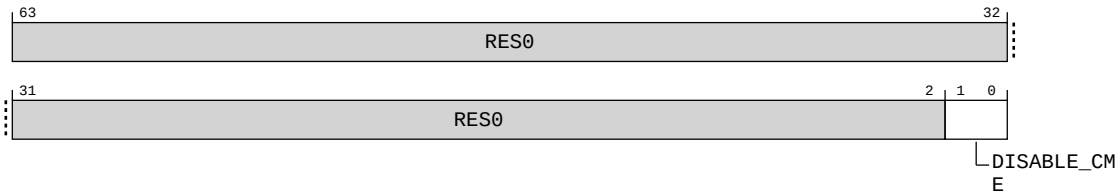


Table A-24: IMP_CABECTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:2]	RES0	Reserved	RES0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[1:0]	DISABLE_CME	**When AArch64-IMP_CMECFR_EL1.NUM_CME == 1** When two SME2 units are implemented (IMP_CMECFR_EL1.NUM_CME == 1), indicate the SME2 units that are not available for assignment **0b00** Both SME2 units are enabled and can be assigned. **0b01** SME2 unit 0 is not available for assignment. **0b10** SME2 unit 1 is not available for assignment. **0b11** Reserved. **When AArch64-IMP_CMECFR_EL1.NUM_CME == 0** **0b00** Both SME2 units are enabled and can be assigned. **0b01** SME2 unit 0 is not available for assignment. **0b10** SME2 unit 1 is not available for assignment. **0b11** Reserved. **Otherwise** RES0	xx

Access

```
MSR S3_0_C15_C12_6, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1100	0b110

```
MRS <Xt>, S3_0_C15_C12_6
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1100	0b110

Accessibility

```
MSR S3_0_C15_C12_6, <Xt>
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.CMEEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.CMEEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CABECTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.CMEEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CABECTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CABECTLR_EL1 = X[t, 64];
```

```
MRS <Xt>, S3_0_C15_C12_6
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CABECTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CABECTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CABECTLR_EL1;
```

A.1.10 IMP_CMERAMDATA0_EL3, Data0 for RAMINDEX

Returns the data from a RAMINDEX instruction using ID=`0x0`,`0x1`.

Configurations

This register is available in all configurations.

Attributes

Width

64

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Functional group

Generic System Control

Access type

RO

Reset value

When CMERAMDATA tag

0000 0000 xxxx xxxx xxxx xxxx x000 0000 0000 0xxx xxxx xxxx xxxx xxxx xxxx xxxx

When CMERAMDATA data

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When CMERAMDATA tag

Returns the tag data from a RAMINDEX instruction

Figure A-10: AARCH64_IMP_CMERAMDATA0_EL3 bit assignments

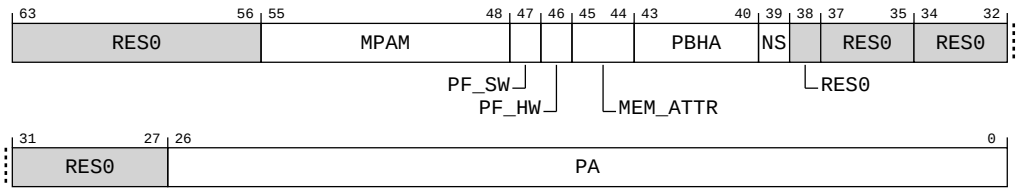


Table A-27: IMP_CMERAMDATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[55:48]	MPAM	MPAM bits	8{x}
[47]	PF_SW	Line brought by SW prefetch	x
[46]	PF_HW	Line brought by HW prefetch	x
[45:44]	MEM_ATTR	Memory attributes	xx
[43:40]	PBHA	PBHA attributes	xxxx
[39]	NS	NS bit	x
[38:27]	**RES0**	Reserved	**RES0**
[26:0]	PA	Physical address [39:13]	27{x}

When CMERAMDATA data

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Returns the data range [63:0] from a RAMINDEX instruction

Figure A-11: AARCH64_IMP_CMERAMDATA0_EL3 bit assignments

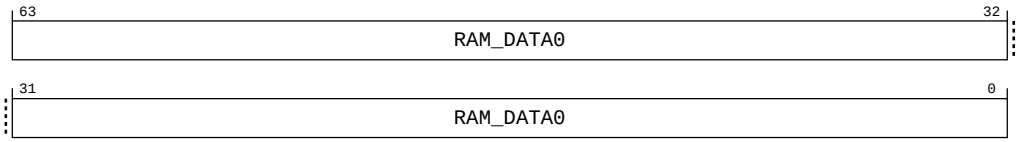


Table A-28: IMP_CMERAMDATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	RAM_DATA0	RAM Data range [63:0]	`64{x}`

Access

`MRS <Xt>, S3_6_C15_C12_2`

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1100	0b010

Accessibility

`MRS <Xt>, S3_6_C15_C12_2`

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMERAMDATA0_EL3;
```

A.1.11 IMP_CMERAMDATA1_EL3, Data1 for RAMINDEX

Returns the data range [127:64] from a RAMINDEX instruction when ID=`0x1`

Configurations

This register is available in all configurations.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

64

Functional group

Generic System Control

Access type

RO

Reset value

When CMERAMDATA tag

0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

When CMERAMDATA data

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When CMERAMDATA tag

Figure A-12: AARCH64_IMP_CMERAMDATA1_EL3 bit assignments

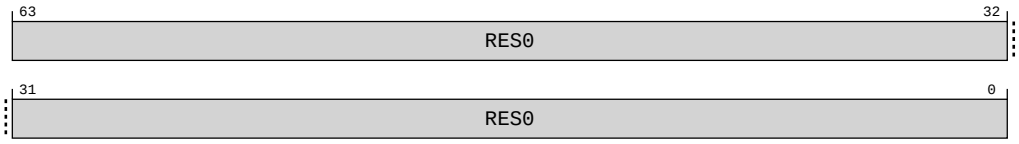


Table A-30: IMP_CMERAMDATA1_EL3 bit descriptions

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

When CMERAMDATA data

Returns the data range [127:64] from a RAMINDEX instruction

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure A-13: AARCH64_IMP_CMERAMDATA1_EL3 bit assignments

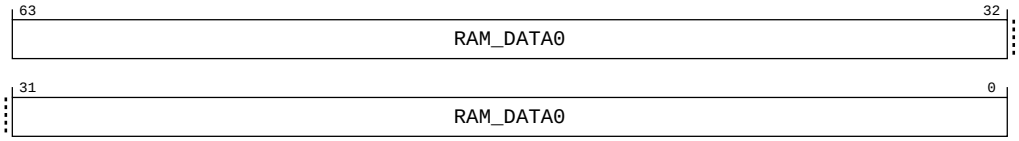


Table A-31: IMP_CMERAMDATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	RAM_DATA0	RAM Data range [127:64]	`64{x}`

Access

`MRS <Xt>, S3_6_C15_C12_3`

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1100	0b011

Accessibility

`MRS <Xt>, S3_6_C15_C12_3`

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMERAMDATA1_EL3;
```

A.1.12 IMP_CMERAMDATA2_EL3, Data2 for RAMINDEX

Returns additional data information from a RAMINDEX instruction

Configurations

This register is available in all configurations.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

64

Functional group

Generic System Control

Access type

RO

Reset value

When CMERAMDATA tag

0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

When CMERAMDATA data

0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When CMERAMDATA tag

Figure A-14: AARCH64_IMP_CMERAMDATA2_EL3 bit assignments

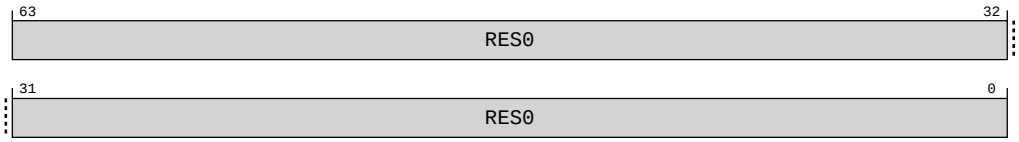


Table A-33: IMP_CMERAMDATA2_EL3 bit descriptions

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

When CMERAMDATA data

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure A-15: AARCH64_IMP_CMERAMDATA2_EL3 bit assignments

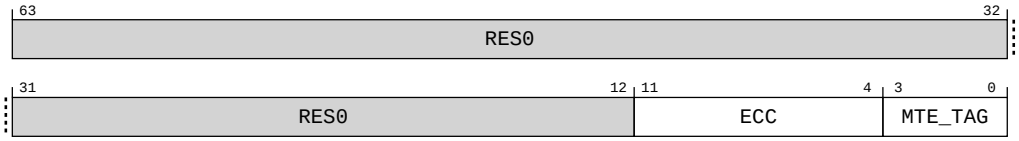


Table A-34: IMP_CMERAMDATA2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:12]	**RES0**	Reserved	**RES0**
[11:4]	ECC	Data ECC	`8{x}`
[3:0]	MTE_TAG	Data MTE Tag	`xxxx`

Access

`MRS <Xt>, S3_6_C15_C12_4`

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1100	0b100

Accessibility

`MRS <Xt>, S3_6_C15_C12_4`

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMERAMDATA2_EL3;
```

A.2 AArch64 Identification registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** Identification registers in the core.

For more information on registers listed in the table, click on the link associated with the register name.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the *Arm® Architecture Reference Manual for A-profile architecture*.

Table A-36: Identification registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SMIDR_EL1	3	1	C0	C0	6	See individual bit resets.	64-bit	Streaming Mode Identification Register
IMP_CMEMPMMCR_EL3	3	6	C15	C12	0	See individual bit resets.	64-bit	Global MPMM Configuration Register

A.2.1 SMIDR_EL1, Streaming Mode Identification Register

Provides additional identification mechanisms for scheduling purposes, for a PE that supports Streaming SVE mode.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

RO

Reset value

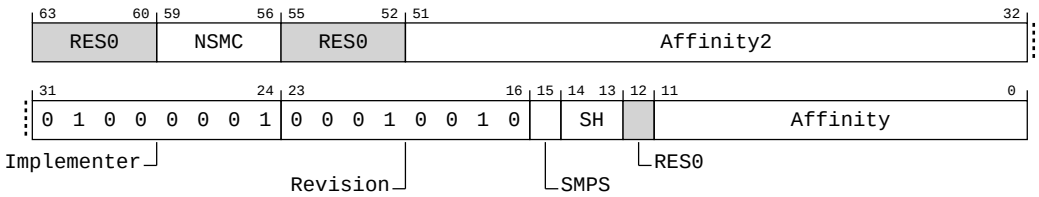
```
0000 xxxx 0000 xxxx xxxx xxxx xxxx xxxx 0100 0001 0001 0010 xxx0 xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-16: AARCH64_SMIDR_EL1 bit assignments



Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table A-37: SMIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	**RES0**	Reserved	**RES0**
[59:56]	NSMC	The number of shared SMCUs usable by this PE minus 1, which correspond to the same values of SMIDR_EL1.Affinity and SMIDR_EL1.Affinity2. • This field is `0b0000` if the implementation of Streaming SVE mode associated with this PE is not shared with other PEs. • The value `0b1111` is reserved for future expansion.	`xxxx`
[55:52]	**RES0**	Reserved	**RES0**
[51:32]	Affinity2	The most significant 20 bits of the SMCU affinity for this PE, to be used in conjunction with SMIDR_EL1.Affinity.	`20{x}`
[31:24]	Implementer	The Implementer code. This field must hold an implementer code that has been assigned by Arm. **`0x41`** Arm Limited.	`0x41`
[23:16]	Revision	Indicates the revision of the product. **`0x12`** r1p2	`0x12`
[15]	SMPS	Indicates support for Streaming SVE mode execution priority. **`0b0`** Priority control not supported. **`0b1`** Priority control supported.	The reset values can be the following: `0b0`, `0b1`, respective to the value.
[14:13]	SH	Indicates whether the implementation of Streaming SVE mode in this PE is shared with other PEs. **`0b11`** The implementation of Streaming SVE mode is shared with other PEs.	`xx`
[12]	**RES0**	Reserved	**RES0**
[11:0]	Affinity	The least significant 12 bits of the SMCU affinity for this PE. The concatenated value SMIDR_EL1.{Affinity2,Affinity} identifies which shared SMCU is associated with the PE. The 32-bit value associated with each SMCU is unique within the system as a whole.	`12{x}`

Access

```
MRS <Xt>, SMIDR_EL1
```

op0	op1	CRn	CRm	op2
`0b11`	`0b001`	`0b0000`	`0b0000`	`0b110`

Accessibility

```
MRS <Xt>, SMIDR_EL1
```

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = SMIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = SMIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SMIDR_EL1;
```

A.2.2 IMP_CMEMPMMCR_EL3, Global MPMM Configuration Register

This register is used to change MPMM gears or disable MPMM.

Configurations

AArch64 register IMP_CMEMPMMCR_EL3 bits [63:0] are architecturally mapped to External register B.2.2 CMEMPMMCR, Global MPMM Configuration Register on page 228 bits [63:0].

Attributes

Width

64

Functional group

Identification registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx11 xxxx xxxx xxxx xxxx xxxx xxxx xxxx x000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure A-17: AARCH64_IMP_CMEMPMMCR_EL3 bit assignments

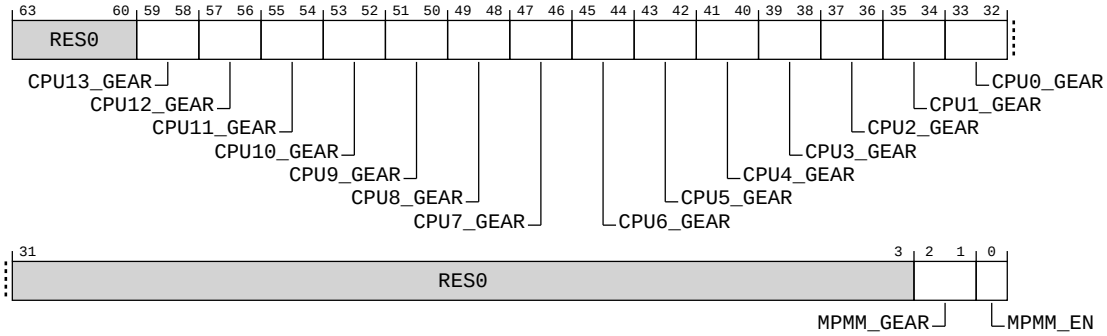


Table A-39: IMP_CMEMPMMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:60]	**RES0**	Reserved	**RES0**
[59:58]	CPU13_GEAR	**When NUM_CORES == 14** Gear for CPU13. The most constraining gear between CPU13_GEAR and MPMM_GEAR is applied to CPU13. **0b00** Maximum selected gear for CPU13 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU13 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU13 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU13 is MPMM_GEAR. **Otherwise** RES0	0b11
[57:56]	CPU12_GEAR	**When NUM_CORES >= 13** Gear for CPU12. The most constraining gear between CPU12_GEAR and MPMM_GEAR is applied to CPU12. **0b00** Maximum selected gear for CPU12 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU12 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU12 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU12 is MPMM_GEAR. **Otherwise** RES0	0b11

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[55:54]	CPU11_GEAR	**When NUM_CORES >= 12** Gear for CPU11. The most constraining gear between CPU11_GEAR and MPMM_GEAR is applied to CPU11. **0b00** Maximum selected gear for CPU11 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU11 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU11 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU11 is MPMM_GEAR. **Otherwise** RES0	0b11
[53:52]	CPU10_GEAR	**When NUM_CORES >= 11** Gear for CPU10. The most constraining gear between CPU10_GEAR and MPMM_GEAR is applied to CPU10. **0b00** Maximum selected gear for CPU10 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU10 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU10 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU10 is MPMM_GEAR. **Otherwise** RES0	0b11
[51:50]	CPU9_GEAR	**When NUM_CORES >= 10** Gear for CPU9. The most constraining gear between CPU9_GEAR and MPMM_GEAR is applied to CPU9. **0b00** Maximum selected gear for CPU9 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU9 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU9 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU9 is MPMM_GEAR. **Otherwise** RES0	0b11

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[49:48]	CPU8_GEAR	**When NUM_CORES >= 9** Gear for CPU8. The most constraining gear between CPU8_GEAR and MPMM_GEAR is applied to CPU8. **0b00** Maximum selected gear for CPU8 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU8 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU8 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU8 is MPMM_GEAR. **Otherwise** RES0	0b11
[47:46]	CPU7_GEAR	**When NUM_CORES >= 8** Gear for CPU7. The most constraining gear between CPU7_GEAR and MPMM_GEAR is applied to CPU7. **0b00** Maximum selected gear for CPU7 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU7 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU7 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU7 is MPMM_GEAR. **Otherwise** RES0	0b11
[45:44]	CPU6_GEAR	**When NUM_CORES >= 7** Gear for CPU6. The most constraining gear between CPU6_GEAR and MPMM_GEAR is applied to CPU6. **0b00** Maximum selected gear for CPU6 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU6 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU6 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU6 is MPMM_GEAR. **Otherwise** RES0	0b11

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[43:42]	CPU5_GEAR	**When NUM_CORES >= 6** Gear for CPU5. The most constraining gear between CPU5_GEAR and MPMM_GEAR is applied to CPU5. **0b00** Maximum selected gear for CPU5 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU5 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU5 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU5 is MPMM_GEAR. **Otherwise** RES0	0b11
[41:40]	CPU4_GEAR	**When NUM_CORES >= 5** Gear for CPU4. The most constraining gear between CPU4_GEAR and MPMM_GEAR is applied to CPU4. **0b00** Maximum selected gear for CPU4 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU4 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU4 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU4 is MPMM_GEAR. **Otherwise** RES0	0b11
[39:38]	CPU3_GEAR	**When NUM_CORES >= 4** Gear for CPU3. The most constraining gear between CPU3_GEAR and MPMM_GEAR is applied to CPU3. **0b00** Maximum selected gear for CPU3 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU3 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU3 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU3 is MPMM_GEAR. **Otherwise** RES0	0b11

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[37:36]	CPU2_GEAR	**When NUM_CORES >= 3** Gear for CPU2. The most constraining gear between CPU2_GEAR and MPMM_GEAR is applied to CPU2. **0b00** Maximum selected gear for CPU2 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU2 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU2 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU2 is MPMM_GEAR. **Otherwise** RES0	0b11
[35:34]	CPU1_GEAR	**When NUM_CORES >= 2** Gear for CPU1. The most constraining gear between CPU1_GEAR and MPMM_GEAR is applied to CPU1. **0b00** Maximum selected gear for CPU1 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU1 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU1 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU1 is MPMM_GEAR. **Otherwise** RES0	0b11
[33:32]	CPU0_GEAR	Gear for CPU0. The most constraining gear between CPU0_GEAR and MPMM_GEAR is applied to CPU0 **0b00** Maximum selected gear for CPU0 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU0 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU0 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU0 is MPMM_GEAR.	0b11
[31:3]	**RES0**	Reserved	**RES0**

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[2:1]	MPMM_GEAR	MPMM Gear Select **0b00** Select MPMM Gear 0. **0b01** Select MPMM Gear 1. **0b10** Select MPMM Gear 2. **0b11** Do not throttle.	0b00
[0]	MPMM_EN	MPMM Enable **0b0** MPMM is not enabled. **0b1** MPMM is enabled.	0b0

Access

MRS <Xt>, S3_6_C15_C12_0

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1100	0b000

MSR S3_6_C15_C12_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1100	0b000

Accessibility

MRS <Xt>, S3_6_C15_C12_0

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMEMPMMCR_EL3;
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
MSR S3_6_C15_C12_0, <Xt>
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CMEMPMMCR_EL3 = X[t, 64];
```

A.3 AArch64 RAS registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** RAS registers in the core.

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the *Arm® Architecture Reference Manual for A-profile architecture*.

Table A-42: RAS registers summary

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

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ERXPFGCDN_EL1	3	0	C5	C4	6	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Countdown Register
ERXMISC0_EL1	3	0	C5	C5	0	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 0
ERXMISC1_EL1	3	0	C5	C5	1	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 1
ERXMISC2_EL1	3	0	C5	C5	2	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 2
ERXMISC3_EL1	3	0	C5	C5	3	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 3

A.3.1 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 registers

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure A-18: AARCH64_ERRIDR_EL1 bit assignments

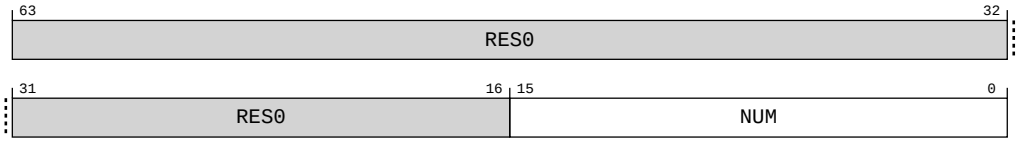


Table A-43: ERRIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15:0]	NUM	Highest numbered index of the records that can be accessed through the Error Record System registers plus one. Zero indicates no records can be accessed through the Error Record System registers. Each implemented record is owned by a node. A node might own multiple records.	16{x}

Access

MRS <Xt>, ERRIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0011	0b000

Accessibility

MRS <Xt>, ERRIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERRIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERRIDR_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERRIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERRIDR_EL1;
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

A.3.2 ERRSELR_EL1, Error Record Select Register

Selects an error record to be accessed through the Error Record System registers.

Configurations

If 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 registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-19: AARCH64_ERRSELR_EL1 bit assignments

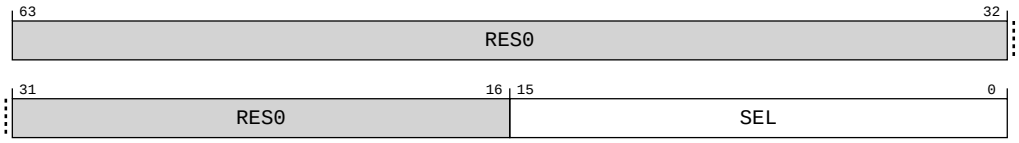


Table A-45: ERRSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[15:0]	SEL	Selects the error record accessed through the ERX registers. For example, if ERRSELR_EL1.SEL is `0x0004`, then direct reads and writes of ERXSTATUS_EL1 access ERR4STATUS. If ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then all of the following apply: • The value read back from ERRSELR_EL1.SEL is **UNKNOWN**. • One of the following occurs: ◦ An **UNKNOWN** error record is selected. ◦ The ERX*_EL1 registers are **RAZ/WI**. ◦ ERX*_EL1 register reads and writes are NOPs. ◦ ERX*_EL1 register reads and writes are **UNDEFINED**.	`16{x}`

Access

```
MRS <Xt>, ERRSELR_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0011	0b001

```
MSR ERRSELR_EL1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0011	0b001

Accessibility

```
MRS <Xt>, ERRSELR_EL1
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERRSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERRSELR_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERRSELR_EL1;
elsif PSTATE.EL == EL3 then
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
    X[t, 64] = ERRSELR_EL1;
```

```
MSR ERRSELR_EL1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif 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
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERRSELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERRSELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERRSELR_EL1 = X[t, 64];
```

A.3.3 ERXFR_EL1, Selected Error Record Feature Register

Accesses ERR<n>FR for the error record <n> selected by ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-20: AARCH64_ERXFR_EL1 bit assignments

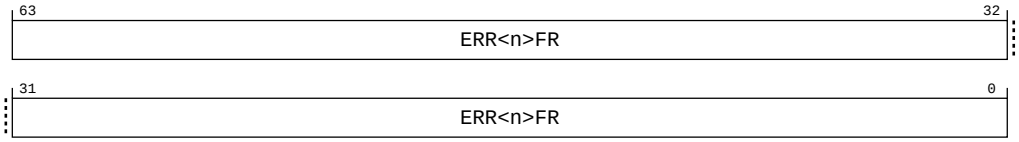


Table A-48: ERXFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXFR_EL1 accesses ERR<n>FR, where <n> is the value in ERRSELR_EL1.SEL.	`64{x}`

Access

```
MRS <Xt>, ERXFR_EL1
```

op0	op1	CRn	CRm	op2
`0b11`	`0b000`	`0b0101`	`0b0100`	`0b000`

Accessibility

If ERRIDR_EL1.NUM is `0x0000` or ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXFR_EL1 is **RAZ**.
- Direct reads of ERXFR_EL1 are NOPs.
- Direct reads of ERXFR_EL1 are **UNDEFINED**.

```
MRS <Xt>, ERXFR_EL1
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
        X[t, 64] = ERXFR_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXFR_EL1;
```

A.3.4 ERXCTLR_EL1, Selected Error Record Control Register

Accesses ERR<n>CTLR for the error record <n> selected by ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

RW

Reset value

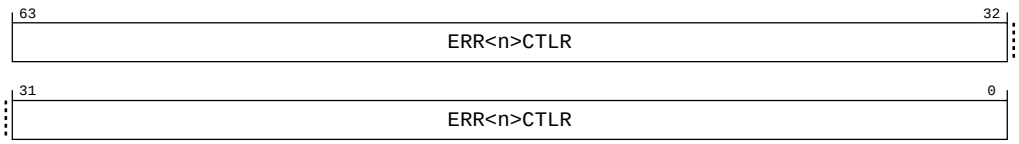
```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-21: AARCH64_ERXCTLR_EL1 bit assignments



Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table A-50: ERXCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXCTLR_EL1 accesses ERR<n>CTLR, where <n> is the value in ERRSELR_EL1.SEL.	`64{x}`

Access

```
MRS <Xt>, ERXCTLR_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b001

```
MSR ERXCTLR_EL1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b001

Accessibility

If ERRIDR_EL1.NUM is `0x0000` or ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXCTLR_EL1 is **RAZ/WI**.
- Direct reads and writes of ERXCTLR_EL1 are NOPs.
- Direct reads and writes of ERXCTLR_EL1 are **UNDEFINED**.

If ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ERR<n>CTLR is not present, meaning reads and writes of ERXCTLR_EL1 are **RES0**.

```
MRS <Xt>, ERXCTLR_EL1
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXCTLR_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXCTLR_EL1;
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXCTLR_EL1;
```

```
MSR ERXCTLR_EL1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif 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
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXCTLR_EL1 = X[t, 64];
```

A.3.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register

Accesses ERR<n>STATUS for the error record <n> selected by ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-22: AARCH64_ERXSTATUS_EL1 bit assignments

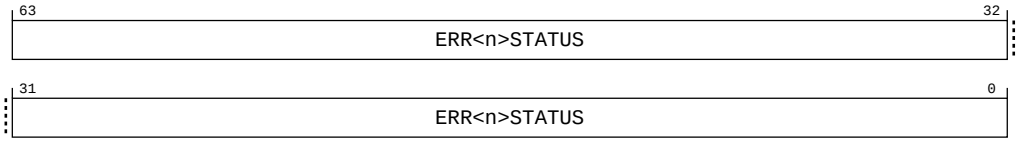


Table A-53: ERXSTATUS_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXSTATUS_EL1 accesses ERR<n>STATUS, where <n> is the value in ERRSELR_EL1.SEL.	`64{x}`

Access

```
MRS <Xt>, ERXSTATUS_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b010

```
MSR ERXSTATUS_EL1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b010

Accessibility

If ERRIDR_EL1.NUM is `0x0000` or ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXSTATUS_EL1 is **RAZ/WI**.
- Direct reads and writes of ERXSTATUS_EL1 are NOPs.
- Direct reads and writes of ERXSTATUS_EL1 are **UNDEFINED**.

ERR<n>STATUS describes additional constraints that also apply when ERR<n>STATUS is accessed through ERXSTATUS_EL1.

```
MRS <Xt>, ERXSTATUS_EL1
```

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

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXSTATUS_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXSTATUS_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXSTATUS_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXSTATUS_EL1;
```

MSR ERXSTATUS_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXSTATUS_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXSTATUS_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXSTATUS_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXSTATUS_EL1 = X[t, 64];
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

A.3.6 ERXADDR_EL1, Selected Error Record Address Register

Accesses ERR<n>ADDR for the error record <n> selected by ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-23: AARCH64_ERXADDR_EL1 bit assignments

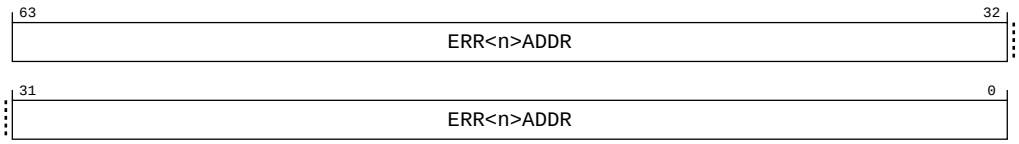


Table A-56: ERXADDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXADDR_EL1 accesses ERR<n>ADDR, where <n> is the value in ERRSELR_EL1.SEL.	`64{x}`

Access

```
MRS <Xt>, ERXADDR_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b011

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
MSR ERXADDR_EL1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b011

Accessibility

If ERRIDR_EL1.NUM is `0x0000` or ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXADDR_EL1 is **RAZ/WI**.
- Direct reads and writes of ERXADDR_EL1 are NOPs.
- Direct reads and writes of ERXADDR_EL1 are **UNDEFINED**.

ERR<n>ADDR describes additional constraints that also apply when ERR<n>ADDR is accessed through ERXADDR_EL1.

```
MRS <Xt>, ERXADDR_EL1
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXADDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXADDR_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXADDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXADDR_EL1;
```

```
MSR ERXADDR_EL1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif 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);
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXADDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXADDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXADDR_EL1 = X[t, 64];
```

A.3.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature Register

Accesses ERR<n>PFGF for the error record <n> selected by ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure A-24: AARCH64_ERXPFGF_EL1 bit assignments

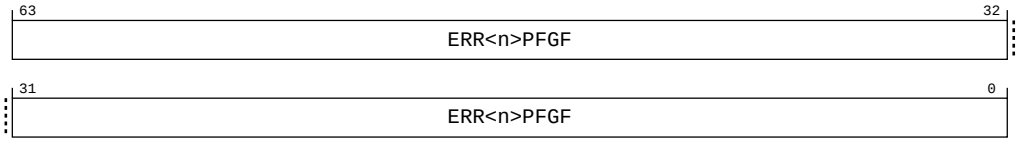


Table A-59: ERXPFGF_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXPFGF_EL1 accesses ERR<n>PFGF, where <n> is the value in ERRSELR_EL1.SEL.	`64{x}`

Access

```
MRS <Xt>, ERXPFGF_EL1
```

op0	op1	CRn	CRm	op2
`0b11`	`0b000`	`0b0101`	`0b0100`	`0b100`

Accessibility

If ERRIDR_EL1.NUM is `0x0000` or ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXPFGF_EL1 is **RAZ**.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are **UNDEFINED**.

If ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGF_EL1 is **RAZ**.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are **UNDEFINED**.

Note

A node does not implement the Common Fault Injection Model Extension if ERR<q>FR.INJ reads as `0b00`. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in ERRSELR_EL1.SEL. If the node owns a single record then q = n.

If ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ERR<n>PFGF is not present, meaning reads of ERXPFGF_EL1 are **RES0**.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

ERR<n>PFGF describes additional constraints that also apply when ERR<n>PFGF is accessed through ERXPFGF_EL1.

`MRS <Xt>, ERXPFGF_EL1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXPFGF_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGF_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif SCR_EL3.FIEN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGF_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXPFGF_EL1;
```

A.3.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control Register

Accesses ERR<n>PFGCTL for the error record <n> selected by ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-25: AARCH64_ERXPFGCTL_EL1 bit assignments

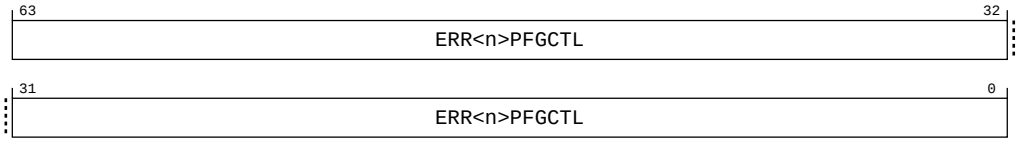


Table A-61: ERXPFGCTL_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXPFGCTL_EL1 accesses ERR<n>PFGCTL, where <n> is the value in ERRSELR_EL1.SEL.	`64{x}`

Access

```
MRS <Xt>, ERXPFGCTL_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b101

```
MSR ERXPFGCTL_EL1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b101

Accessibility

If ERRIDR_EL1.NUM is `0x0000` or ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXPFGCTL_EL1 is **RAZ/WI**.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are **UNDEFINED**.

If ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCTL_EL1 is **RAZ/WI**.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are **UNDEFINED**.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

A node does not implement the Common Fault Injection Model Extension if ERR<q>FR.INJ reads as `0b00`. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in ERRSELR_EL1.SEL. If the node owns a single record then q = n.

If ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ERR<n>PFGCTL is not present, meaning reads and writes of ERXPFGCTL_EL1 are **RES0**.

ERR<n>PFGCTL describes additional constraints that also apply when ERR<n>PFGCTL is accessed through ERXPFGCTL_EL1.

`MRS <Xt>, ERXPFGCTL_EL1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXPFGCTL_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGCTL_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif SCR_EL3.FIEN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGCTL_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXPFGCTL_EL1;
```

`MSR ERXPFGCTL_EL1, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXPFGCTL_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCTL_EL1 = X[t, 64];
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif SCR_EL3.FIEN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCTL_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXPFGCTL_EL1 = X[t, 64];
```

A.3.9 ERXPFGCDN_EL1, Selected Pseudo-fault Generation Countdown Register

Accesses ERR<n>PFGCDN for the error record <n> selected by ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

RW

Reset value

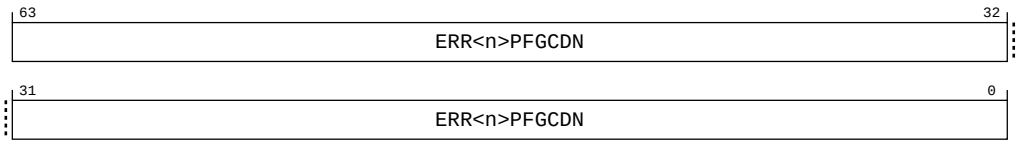
```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-26: AARCH64_ERXPFGCDN_EL1 bit assignments



Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table A-64: ERXPFGCDN_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXPFGCDN_EL1 accesses ERR<n>PFGCDN, where <n> is the value in ERRSELR_EL1.SEL.	`64{x}`

Access

```
MRS <Xt>, ERXPFGCDN_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b110

```
MSR ERXPFGCDN_EL1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b110

Accessibility

If ERRIDR_EL1.NUM is `0x0000` or ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXPFGCDN_EL1 is **RAZ/WI**.
- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are **UNDEFINED**.

If ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCDN_EL1 is **RAZ/WI**.
- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are **UNDEFINED**.

Note

A node does not implement the Common Fault Injection Model Extension if ERR<q>FR.INJ reads as `0b00`. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in ERRSELR_EL1.SEL. If the node owns a single record then q = n.

If ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ERR<n>PFGCDN is not present, meaning reads and writes of ERXPFGCDN_EL1 are **RES0**.

ERR<n>PFGCDN describes additional constraints that also apply when ERR<n>PFGCDN is accessed through ERXPFGCDN_EL1.

```
MRS <Xt>, ERXPFGCDN_EL1
```

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

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXPFGCDN_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGCDN_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif SCR_EL3.FIEN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGCDN_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXPFGCDN_EL1;
```

MSR ERXPFGCDN_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif 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
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif SCR_EL3.FIEN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXPFGCDN_EL1 = X[t, 64];
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

A.3.10 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0

Accesses ERR<n>MISC0 for the error record <n> selected by ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-27: AARCH64_ERXMISC0_EL1 bit assignments

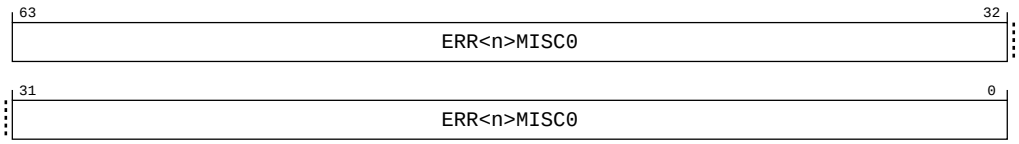


Table A-67: ERXMISC0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXMISC0_EL1 accesses ERR<n>MISC0, where <n> is the value in ERRSELR_EL1.SEL.	`64{x}`

Access

```
MRS <Xt>, ERXMISC0_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b000

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
MSR ERXMISC0_EL1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b000

Accessibility

If ERRIDR_EL1.NUM is `0x0000` or ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC0_EL1 is **RAZ/WI**.
- Direct reads and writes of ERXMISC0_EL1 are NOPs.
- Direct reads and writes of ERXMISC0_EL1 are **UNDEFINED**.

ERR<n>MISC0 describes additional constraints that also apply when ERR<n>MISC0 is accessed through ERXMISC0_EL1.

```
MRS <Xt>, ERXMISC0_EL1
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC0_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXMISC0_EL1;
```

```
MSR ERXMISC0_EL1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC0_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC0_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXMISC0_EL1 = X[t, 64];
```

A.3.11 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1

Accesses ERR<n>MISC1 for the error record <n> selected by ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure A-28: AARCH64_ERXMISC1_EL1 bit assignments

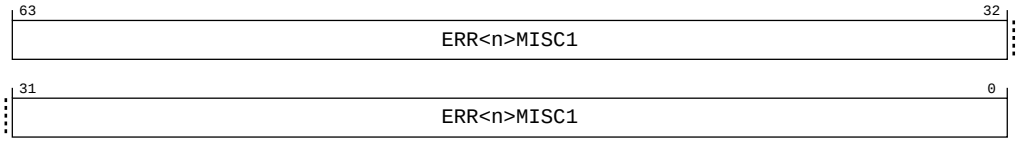


Table A-70: ERXMISC1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXMISC1_EL1 accesses ERR<n>MISC1, where <n> is the value in ERRSELR_EL1.SEL.	`64{x}`

Access

`MRS <Xt>, ERXMISC1_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b001

`MSR ERXMISC1_EL1, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b001

Accessibility

If ERRIDR_EL1.NUM is `0x0000` or ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC1_EL1 is **RAZ/WI**.
- Direct reads and writes of ERXMISC1_EL1 are NOPs.
- Direct reads and writes of ERXMISC1_EL1 are **UNDEFINED**.

ERR<n>MISC1 describes additional constraints that also apply when ERR<n>MISC1 is accessed through ERXMISC1_EL1.

`MRS <Xt>, ERXMISC1_EL1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC1_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXMISC1_EL1;
```

```
MSR ERXMISC1_EL1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif 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
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC1_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC1_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXMISC1_EL1 = X[t, 64];
```

A.3.12 ERXMISC2_EL1, Selected Error Record Miscellaneous Register 2

Accesses ERR<n>MISC2 for the error record <n> selected by ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Functional group

RAS registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-29: AARCH64_ERXMISC2_EL1 bit assignments

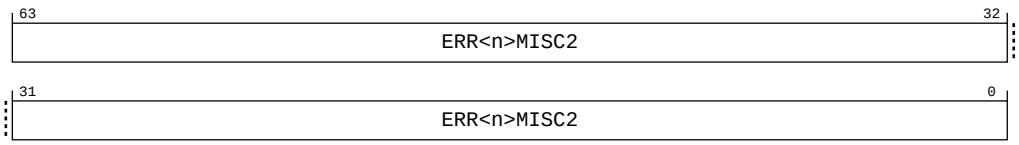


Table A-73: ERXMISC2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXMISC2_EL1 accesses ERR<n>MISC2, where <n> is the value in ERRSELR_EL1.SEL.	`64{x}`

Access

```
MRS <Xt>, ERXMISC2_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b010

```
MSR ERXMISC2_EL1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b010

Accessibility

If ERRIDR_EL1.NUM is `0x0000` or ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC2_EL1 is **RAZ/WI**.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

- Direct reads and writes of ERXMISC2_EL1 are NOPs.
- Direct reads and writes of ERXMISC2_EL1 are **UNDEFINED**.

ERR<n>MISC2 describes additional constraints that also apply when ERR<n>MISC2 is accessed through ERXMISC2_EL1.

`MRS <Xt>, ERXMISC2_EL1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC2_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXMISC2_EL1;
```

`MSR ERXMISC2_EL1, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif 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
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
    else
        ERXMISC2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXMISC2_EL1 = X[t, 64];
```

A.3.13 ERXMISC3_EL1, Selected Error Record Miscellaneous Register 3

Accesses ERR<n>MISC3 for the error record <n> selected by ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-30: AARCH64_ERXMISC3_EL1 bit assignments

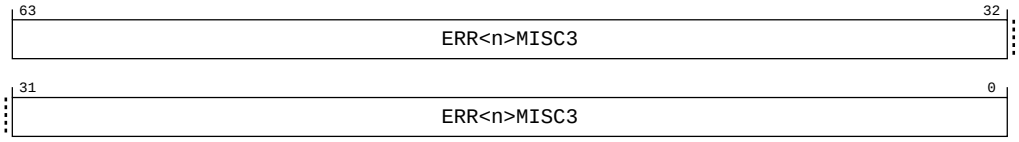


Table A-76: ERXMISC3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXMISC3_EL1 accesses ERR<n>MISC3, where <n> is the value in ERRSELR_EL1.SEL.	`64{x}`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Access

`MRS <Xt>, ERXMISC3_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b011

`MSR ERXMISC3_EL1, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b011

Accessibility

If ERRIDR_EL1.NUM is `0x0000` or ERRSELR_EL1.SEL is greater than or equal to ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC3_EL1 is **RAZ/WI**.
- Direct reads and writes of ERXMISC3_EL1 are NOPs.
- Direct reads and writes of ERXMISC3_EL1 are **UNDEFINED**.

ERR<n>MISC3 describes additional constraints that also apply when ERR<n>MISC3 is accessed through ERXMISC3_EL1.

`MRS <Xt>, ERXMISC3_EL1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC3_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC3_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXMISC3_EL1;
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
MSR ERXMISC3_EL1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif 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
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC3_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC3_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXMISC3_EL1 = X[t, 64];
```

A.4 AArch64 Special-purpose registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** Special-purpose registers in the core.

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the *Arm® Architecture Reference Manual for A-profile architecture*.

Table A-79: Special-purpose registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CMEPPMCR_EL3	3	6	C15	C11	7	See individual bit resets.	64-bit	Global PPM Configuration Register
IMP_CMEMPMMTUNE_EL3	3	6	C15	C12	1	See individual bit resets.	64-bit	MPMM Tuning Configuration Register

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

A.4.1 IMP_CMEPPMCR_EL3, Global PPM Configuration Register

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

Configurations

AArch64 register IMP_CMEPPMCR_EL3 bits [63:0] are architecturally mapped to External register B.2.1 CMEPPMCR, Global PPM Configuration Register on page 227 bits [63:0].

Attributes

Width

64

Functional group

Special-purpose registers

Access type

RW

Reset value

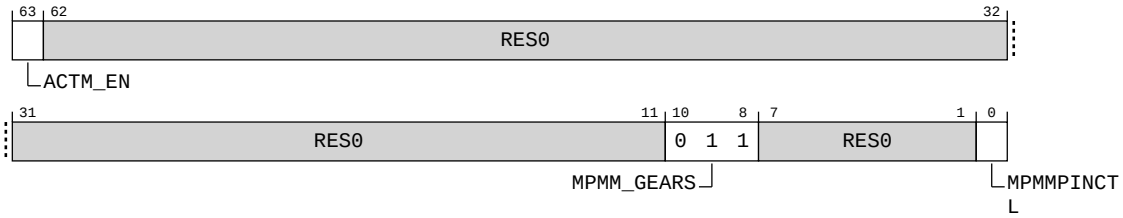
```
0xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x011 xxxx xxx0
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-31: AARCH64_IMP_CMEPPMCR_EL3 bit assignments



Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table A-80: IMP_CMEPPMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63]	ACTM_EN	Activity Meter Enable **0b0** Disable Activity Meter logic pins. **0b1** Enable Activity Meter logic pins.	0b0
[62:11]	**RES0**	Reserved	**RES0**
[10:8]	MPMM_GEARS	Number of MPMM Gears implemented **0b011** 3 MPMM gears are available.	0b011
[7:1]	**RES0**	Reserved	**RES0**
[0]	MPMMPINCTL	MPMM Pin Control Enabled **0b0** MPMM control through SPR and utility bus. **0b1** MPMM control through pin only.	0b0

Access

```
MRS <Xt>, S3_6_C15_C11_7
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1011	0b111

```
MSR S3_6_C15_C11_7, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1011	0b111

Accessibility

```
MRS <Xt>, S3_6_C15_C11_7
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
    X[t, 64] = IMP_CMEPPMCR_EL3;
```

```
MSR S3_6_C15_C11_7, <Xt>
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CMEPPMCR_EL3 = X[t, 64];
```

A.4.2 IMP_CMEMPMMTUNE_EL3, MPMM Tuning Configuration Register

This register contains the fine tuning/trim configuration for MPMM.

Configurations

AArch64 register IMP_CMEMPMMTUNE_EL3 bits [63:0] are architecturally mapped to External register B.2.3 CMEMPMMTUNE, MPMM Tuning Configuration Register on page 235 bits [63:0].

Attributes

Width

64

Functional group

Special-purpose registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

This register contains control bits that affect the CPU behavior

Figure A-32: AARCH64_IMP_CMEMPMMTUNE_EL3 bit assignments

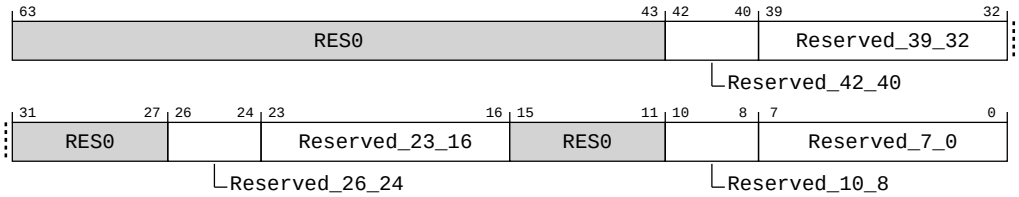


Table A-83: IMP_CMEMPMMTUNE_EL3 bit descriptions

Bits	Name	Description	Reset
[63:43]	**RES0**	Reserved	**RES0**
[42:40]	Reserved_42_40	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_42_40 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_42_40 **Otherwise** RES0	`0b001`
[39:32]	Reserved_39_32	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_39_32 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_39_32 **Otherwise** RES0	The reset values can be the following: `0x83`, `0x85`, respective to the value
[31:27]	**RES0**	Reserved	**RES0**
[26:24]	Reserved_26_24	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_26_24 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_26_24 **Otherwise** RES0	`0b010`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[23:16]	Reserved_23_16	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_23_16 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_23_16 **Otherwise** RES0	The reset values can be the following: `0x6B`, `0x70`, respective to the value
[15:11]	**RES0**	Reserved	**RES0**
[10:8]	Reserved_10_8	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_10_8 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_10_8 **Otherwise** RES0	`0b011`
[7:0]	Reserved_7_0	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_7_0 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_7_0 **Otherwise** RES0	`0x50`

Access

```
MRS <Xt>, S3_6_C15_C12_1
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1100	0b001

```
MSR S3_6_C15_C12_1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1100	0b001

Accessibility

```
MRS <Xt>, S3_6_C15_C12_1
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CMEMPMMTUNE_EL3;
```

```
MSR S3_6_C15_C12_1, <Xt>
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CMEMPMMTUNE_EL3 = X[t, 64];
```

A.5 AArch64 System instructions summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** System instructions in the core.

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the *Arm® Architecture Reference Manual for A-profile architecture*.

Table A-86: System instructions summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SYS_IMP_CMERAMINDEX	1	6	C15	C12	0	See individual bit resets.	64-bit	SYS_IMP_CMERAMINDEX system instruction

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

A.5.1 SYS_IMP_CMERAMINDEX, SYS_IMP_CMERAMINDEX system instruction

This instruction reads data from the selected memory. Data are visible in IMP_CMERAMDATA0_EL3, IMP_CMERAMDATA1_EL3 and IMP_CMERAMDATA2_EL3

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Bit descriptions

When UInt(AArch64-SYS_IMP_CMERAMINDEX.ID) == 0x0

Figure A-33: AARCH64_SYS_IMP_CMERAMINDEX bit assignments

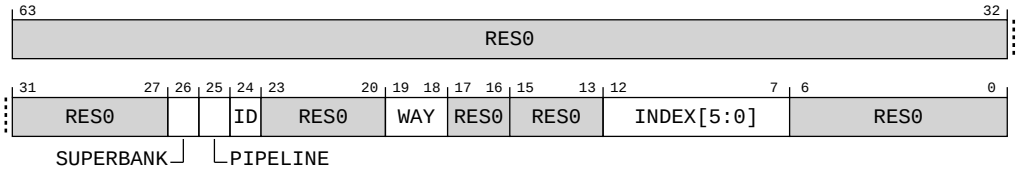


Table A-87: SYS_IMP_CMERAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:27]	**RES0**	Reserved	**RES0**
[26]	SUPERBANK	Superbank number selection SUPERBANK[26] value is composed by PA[6]	x
[25]	PIPELINE	Pipeline number in the selected Superbank PIPELINE[25] value is composed by Hash[1], with the hashing function being Hash[6:0] = PA[12:6] + PA[19:13] + {PA[16:13], 3'b000}	x
[24]	ID	Select the RAM in the Pipeline **0b0** Tagram	x
[23:20]	**RES0**	Reserved	**RES0**
[19:18]	WAY	Way of the selected RAM	xx
[17:13]	**RES0**	Reserved	**RES0**

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[12:7]	INDEX[5:0]	Index selection for Tagram INDEX[5:0] value is composed by {Hash[6:2],Hash[0]}, with the hashing function being Hash[6:0] = PA[12:6] + PA[19:13] + {PA[16:13], 3'b000}	`6{x}`
[6:0]	**RES0**	Reserved	**RES0**

When UInt(AArch64-SYS_IMP_CMERAMINDEX.ID) == 0x1

Figure A-34: AARCH64_SYS_IMP_CMERAMINDEX bit assignments

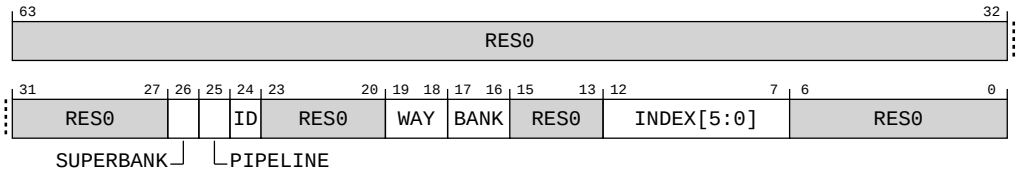


Table A-88: SYS_IMP_CMERAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:27]	**RES0**	Reserved	**RES0**
[26]	SUPERBANK	Superbank number selection SUPERBANK[26] value is composed by PA[6]	`x`
[25]	PIPELINE	Pipeline number in the selected Superbank PIPELINE[25] value is composed by Hash[1], with the hashing function being Hash[6:0] = PA[12:6] + PA[19:13] + {PA[16:13], 3'b000}	`x`
[24]	ID	Select the RAM in the Pipeline **`0b1`** Dataram	`x`
[23:20]	**RES0**	Reserved	**RES0**
[19:18]	WAY	Way of the selected RAM	`xx`
[17:16]	BANK	BANK for Dataram	`xx`
[15:13]	**RES0**	Reserved	**RES0**
[12:7]	INDEX[5:0]	Index selection for Dataram INDEX[5:0] value is composed by {Hash[6:2],Hash[0]}, with the hashing function being Hash[6:0] = PA[12:6] + PA[19:13] + {PA[16:13], 3'b000}	`6{x}`
[6:0]	**RES0**	Reserved	**RES0**

Access

```
SYS #6, C15, C12, #0{, <Xt>}
```

op0	op1	CRn	CRm	op2
`0b01`	`0b110`	`0b1111`	`0b1100`	`0b000`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

```
SYS #6, C15, C12, #0{, <Xt>}
```

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CMERAMINDEX(X[t, 64]);
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Appendix B External registers

This appendix contains the descriptions for the C1-SME2 external registers.

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

B.1 External AMU registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped AMU registers in the core.

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the *Arm® Architecture Reference Manual for A-profile architecture*.

Table B-1: AMU registers summary

Offset	Name	Reset	Width	Description
0x0	AMEVCNTR00 [63:0]	`0x0000000000000000`	64-bit	Activity Monitors Event Counter Registers 0
0x8	AMEVCNTR01 [63:0]	`0x0000000000000000`	64-bit	Activity Monitors Event Counter Registers 0
0x10	AMEVCNTR02 [63:0]	`0x0000000000000000`	64-bit	Activity Monitors Event Counter Registers 0
0x18	AMEVCNTR03 [63:0]	`0x0000000000000000`	64-bit	Activity Monitors Event Counter Registers 0
0x100	AMEVCNTR10 [63:0]	`0x0000000000000000`	64-bit	Activity Monitors Event Counter Registers 1
0x108	AMEVCNTR11 [63:0]	`0x0000000000000000`	64-bit	Activity Monitors Event Counter Registers 1
0x110	AMEVCNTR12 [63:0]	`0x0000000000000000`	64-bit	Activity Monitors Event Counter Registers 1
0x118	AMEVCNTR13 [63:0]	`0x0000000000000000`	64-bit	Activity Monitors Event Counter Registers 1
0x400	AMEVTYPER00	`0x00003246`	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	`0x00004004`	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	`0x00003247`	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	`0x0000324F`	32-bit	Activity Monitors Event Type Registers 0
0x480	AMEVTYPER10	`0x00000300`	32-bit	Activity Monitors Event Type Registers 1
0x484	AMEVTYPER11	`0x00000301`	32-bit	Activity Monitors Event Type Registers 1
0x488	AMEVTYPER12	`0x00000302`	32-bit	Activity Monitors Event Type Registers 1
0x48C	AMEVTYPER13	`0x00000310`	32-bit	Activity Monitors Event Type Registers 1
0xC00	AMCNTENSET0	See individual bit resets.	32-bit	Activity Monitors Count Enable Set Register 0
0xC04	AMCNTENSET1	See individual bit resets.	32-bit	Activity Monitors Count Enable Set Register 1
0xC20	AMCNTENCLR0	See individual bit resets.	32-bit	Activity Monitors Count Enable Clear Register 0
0xC24	AMCNTENCLR1	See individual bit resets.	32-bit	Activity Monitors Count Enable Clear Register 1
0xCE0	AMCGCR	`0x00000404`	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	`0x11003F07`	32-bit	Activity Monitors Configuration Register

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Offset	Name	Reset	Width	Description
0xE04	AMCR	See individual bit resets.	32-bit	Activity Monitors Control Register
0xE08	AMIIDR	`0xD8D1243B`	32-bit	Activity Monitors Implementation Identification Register
0xFA8	AMDEVAFF0	`0x00000000`	32-bit	Activity Monitors Device Affinity Register 0
0xFAC	AMDEVAFF1	`0x00000000`	32-bit	Activity Monitors Device Affinity Register 1
0xFBC	AMDEVARCH	`0x47700A66`	32-bit	Activity Monitors Device Architecture Register
0xFCC	AMDEVTYPE	`0x00000016`	32-bit	Activity Monitors Device Type Register
0xFD0	AMPIDR4	`0x00000004`	32-bit	Activity Monitors Peripheral Identification Register 4
0xFE0	AMPIDR0	`0x0000008D`	32-bit	Activity Monitors Peripheral Identification Register 0
0xFE4	AMPIDR1	`0x000000BD`	32-bit	Activity Monitors Peripheral Identification Register 1
0xFE8	AMPIDR2	`0x0000001B`	32-bit	Activity Monitors Peripheral Identification Register 2
0xFEC	AMPIDR3	`0x00000020`	32-bit	Activity Monitors Peripheral Identification Register 3
0xFF0	AMCIDR0	`0x0000000D`	32-bit	Activity Monitors Component Identification Register 0
0xFF4	AMCIDR1	`0x00000090`	32-bit	Activity Monitors Component Identification Register 1
0xFF8	AMCIDR2	`0x00000005`	32-bit	Activity Monitors Component Identification Register 2
0xFFC	AMCIDR3	`0x000000B1`	32-bit	Activity Monitors Component Identification Register 3

B.1.1 AMEVCNTR00, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counter 0.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offset

0x0

Access type

RW

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-1: AMU_AMEVCNTR00 bit assignments

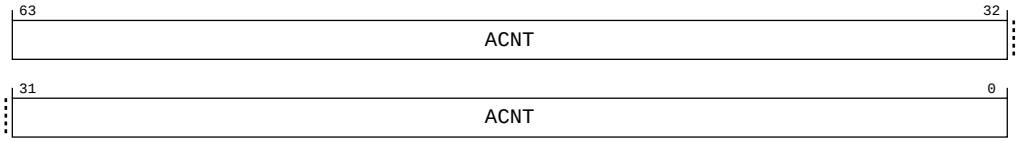


Table B-2: AMEVCNTR00 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of architected activity monitor event counter 0.	0x0000000000000000

Accessibility

If 0 is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR00 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x0	AMEVCNTR00	63:0

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.2 AMEVCNTR01, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counter 1.

Configurations

This register is available in all configurations.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

64

Component

AMU

Register offset

0x8

Access type

RW

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-2: AMU_AMEVCNTR01 bit assignments

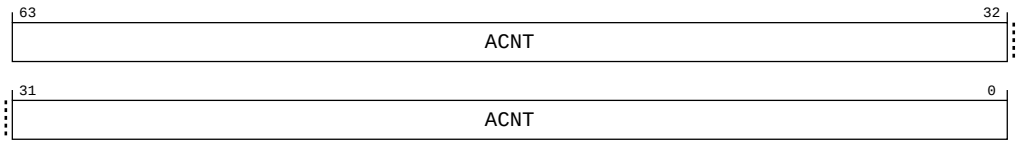


Table B-4: AMEVCNTR01 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of architected activity monitor event counter 1.	0x0000000000000000

Accessibility

If 1 is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR01 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x8	AMEVCNTR01	63:0

This interface is accessible as follows:

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.3 AMEVCNTR02, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counter 2.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offset

0x10

Access type

RW

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-3: AMU_AMEVCNTR02 bit assignments

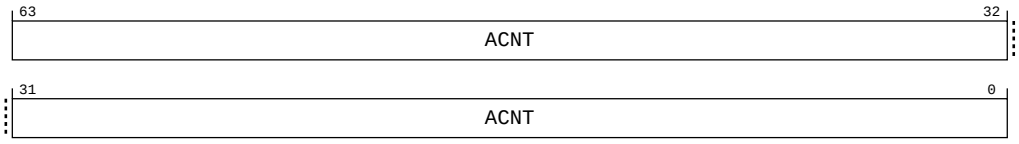


Table B-6: AMEVCNTR02 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of architected activity monitor event counter 2.	0x0000000000000000

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

If 2 is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR02 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x10	AMEVCNTR02	63:0

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.4 AMEVCNTR03, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counter 3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offset

0x18

Access type

RW

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-4: AMU_AMEVCNTR03 bit assignments

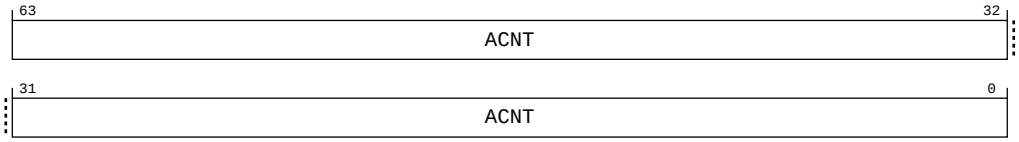


Table B-8: AMEVCNTR03 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of architected activity monitor event counter 3.	0x0000000000000000

Accessibility

If 3 is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR03 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x18	AMEVCNTR03	63:0

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.5 AMEVCNTR10, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counter 0.

Configurations

This register is available in all configurations.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

64

Component

AMU

Register offset

0x100

Access type

RW

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-5: AMU_AMEVCNTR10 bit assignments

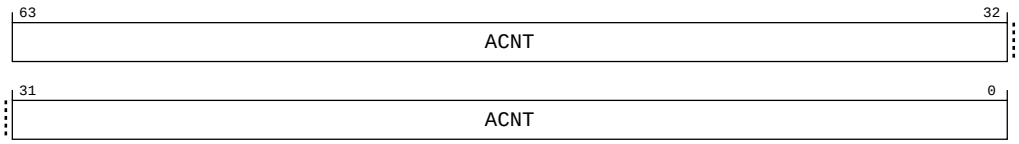


Table B-10: AMEVCNTR10 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of auxiliary activity monitor event counter 0.	0x0000000000000000

Accessibility

If 0 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR10 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x100	AMEVCNTR10	63:0

This interface is accessible as follows:

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.6 AMEVCNTR11, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counter 1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offset

0x108

Access type

RW

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-6: AMU_AMEVCNTR11 bit assignments

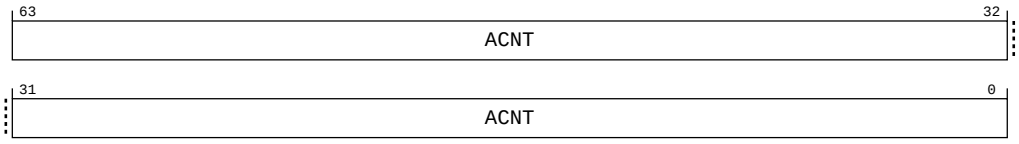


Table B-12: AMEVCNTR11 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of auxiliary activity monitor event counter 1.	0x0000000000000000

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

If 1 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR11 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x108	AMEVCNTR11	63:0

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.7 AMEVCNTR12, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counter 2.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offset

0x110

Access type

RW

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-7: AMU_AMEVCNTR12 bit assignments

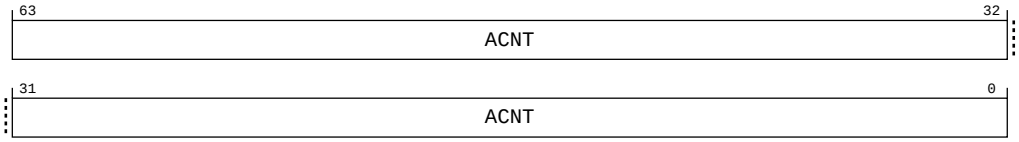


Table B-14: AMEVCNTR12 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of auxiliary activity monitor event counter 2.	0x0000000000000000

Accessibility

If 2 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR12 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x110	AMEVCNTR12	63:0

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.8 AMEVCNTR13, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counter 3.

Configurations

This register is available in all configurations.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

64

Component

AMU

Register offset

0x118

Access type

RW

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-8: AMU_AMEVCNTR13 bit assignments

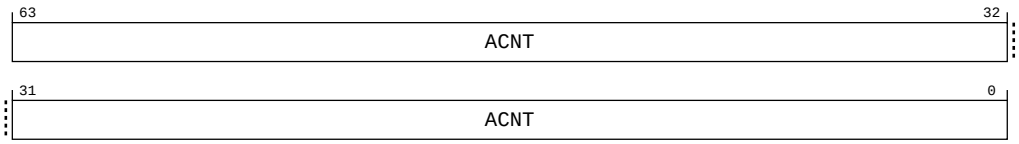


Table B-16: AMEVCNTR13 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of auxiliary activity monitor event counter 3.	0x0000000000000000

Accessibility

If 3 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR13 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x118	AMEVCNTR13	63:0

This interface is accessible as follows:

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.9 AMEVTYPER00, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMEVCNTR00 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x400

Access type

RO

Reset value

```
0000 0000 0000 0000 0011 0010 0100 0110
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-9: AMU_AMEVTYPER00 bit assignments

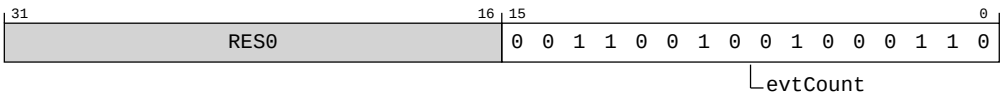


Table B-18: AMEVTYPER00 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AMU.AMEVCNTR00. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0x3246** SME2 unit clock cycles.	0x3246

Accessibility

If 0 is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER00 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x400	AMEVTYPER00	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.10 AMEVTYPER01, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMEVCNTR01 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Register offset

0x404

Access type

RO

Reset value

```
0000 0000 0000 0000 0100 0000 0000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-10: AMU_AMEVTYPER01 bit assignments

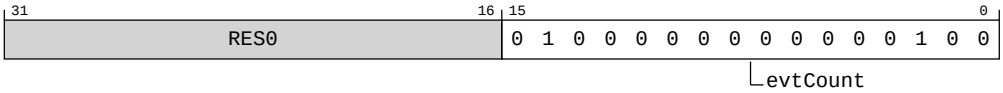


Table B-20: AMEVTYPER01 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AMU.AMEVCNTR01. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0x4004** Constant frequency cycles.	0x4004

Accessibility

If 1 is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER01 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x404	AMEVTYPER01	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.11 AMEVTYPER02, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMEVCNTR02 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x408

Access type

RO

Reset value

```
0000 0000 0000 0000 0011 0010 0100 0111
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-11: AMU_AMEVTYPER02 bit assignments

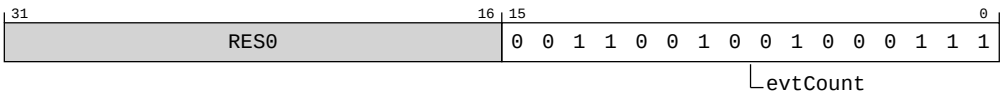


Table B-22: AMEVTYPER02 bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AMU.AMEVCNTR02. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **`0x3247`** Number of instructions retired executed by the SME2 unit.	`0x3247`

Accessibility

If 2 is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER02 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x408	AMEVTYPER02	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.12 AMEVTYPER03, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMEVCNTR03 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Register offset

0x40C

Access type

RO

Reset value

```
0000 0000 0000 0000 0011 0010 0100 1111
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-12: AMU_AMEVTYPER03 bit assignments

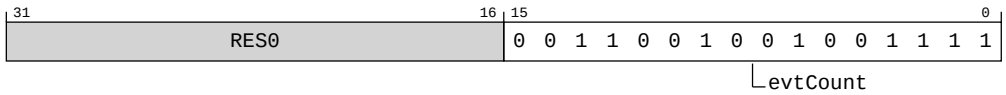


Table B-24: AMEVTYPER03 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AMU.AMEVCNTR03. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0x324F** Stall cycles due to memory backpressure.	0x324F

Accessibility

If 3 is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER03 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x40C	AMEVTYPER03	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.13 AMEVTYPER10, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMEVCNTR10 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x480

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0011 0000 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-13: AMU_AMEVTYPER10 bit assignments

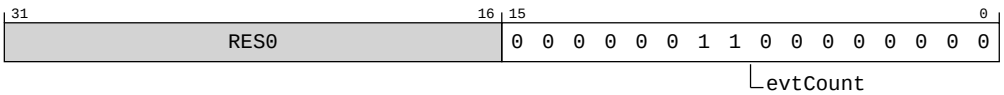


Table B-26: AMEVTYPER10 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AMU.AMEVCNTR10. The following table shows the mapping between required event numbers and the corresponding counters: **0x0300** MPMM gear 0 period threshold exceeded	0x0300

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

If 0 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER10 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x480	AMEVTYPER10	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.14 AMEVTYPER11, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMEVCNTR11 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x484

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0011 0000 0001
|    |    |    |    |    |    |    |  |
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
31    27    23    19    15    11    7     3  0
```

Bit descriptions

Figure B-14: AMU_AMEVTYPER11 bit assignments

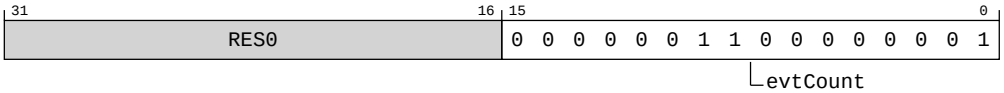


Table B-28: AMEVTYPER11 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AMU.AMEVCNTR11. The following table shows the mapping between required event numbers and the corresponding counters: **0x0301** MPMM gear 1 period threshold exceeded	0x0301

Accessibility

If 1 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER11 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x484	AMEVTYPER11	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.1.15 AMEVTYPER12, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMEVCNTR12 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x488

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0011 0000 0010
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-15: AMU_AMEVTYPER12 bit assignments

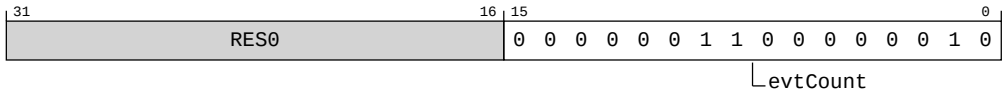


Table B-30: AMEVTYPER12 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AMU.AMEVCNTR12. The following table shows the mapping between required event numbers and the corresponding counters: **0x0302** MPMM gear 2 period threshold exceeded	0x0302

Accessibility

If 2 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER12 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

AMU.AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x488	AMEVTYPER12	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.16 AMEVTYPER13, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMEVCNTR13 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x48C

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0011 0001 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-16: AMU_AMEVTYPER13 bit assignments

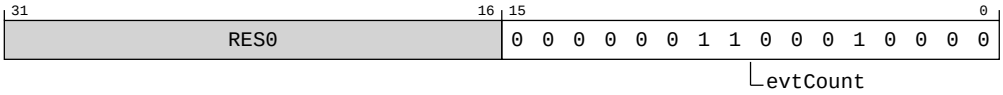


Table B-32: AMEVTYPER13 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AMU.AMEVCNTR13. The following table shows the mapping between required event numbers and the corresponding counters: **0x0310** MPMM Activity Monitor	0x0310

Accessibility

If 3 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER13 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

AMU.AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x48C	AMEVTYPER13	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.1.17 AMCNTENSET0, Activity Monitors Count Enable Set Register 0

Enable control bits for the architected activity monitors event counters, AMU.AMEVCNTR0<n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xC00

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx 0000 0000 0000 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-17: AMU_AMCNTENSET0 bit assignments

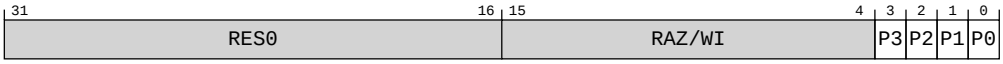


Table B-34: AMCNTENSET0 bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0
[15:4]	RAZ/WI	Reserved	RAZ/WI

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[3]	P3	Activity monitor event counter enable bit for AMU.AMEVCNTR03. **Note:** AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4. Possible values of each bit are: **0b0** When read, means that AMEVCNTR03 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR03 is enabled. When written, enables AMEVCNTR03.	0b0
[2]	P2	Activity monitor event counter enable bit for AMU.AMEVCNTR02. **Note:** AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4. Possible values of each bit are: **0b0** When read, means that AMEVCNTR02 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR02 is enabled. When written, enables AMEVCNTR02.	0b0
[1]	P1	Activity monitor event counter enable bit for AMU.AMEVCNTR01. **Note:** AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4. Possible values of each bit are: **0b0** When read, means that AMEVCNTR01 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR01 is enabled. When written, enables AMEVCNTR01.	0b0
[0]	P0	Activity monitor event counter enable bit for AMU.AMEVCNTR00. **Note:** AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4. Possible values of each bit are: **0b0** When read, means that AMEVCNTR00 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR00 is enabled. When written, enables AMEVCNTR00.	0b0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

Component	Offset	Instance	Range
AMU	0xC00	AMCNTENSET0	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.18 AMCNTENSET1, Activity Monitors Count Enable Set Register 1

Enable control bits for the auxiliary activity monitors event counters, AMU.AMEVCNTR1<n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xC04

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx 0000 0000 0000 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-18: AMU_AMCNTENSET1 bit assignments

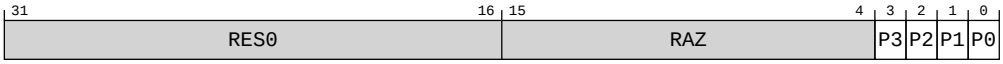


Table B-36: AMCNTENSET1 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:4]	**RAZ**	Reserved	**RAZ**
[3]	P3	Activity monitor event counter enable bit for AMU.AMEVCNTR13. When N is less than 16, bits [15:N] are **RAZ**, where N is the value in AMU.AMCGCR.CG1NC. Possible values of each bit are: **0b0** When read, means that AMEVCNTR13 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR13 is enabled. When written, enables AMEVCNTR13.	0b0
[2]	P2	Activity monitor event counter enable bit for AMU.AMEVCNTR12. When N is less than 16, bits [15:N] are **RAZ**, where N is the value in AMU.AMCGCR.CG1NC. Possible values of each bit are: **0b0** When read, means that AMEVCNTR12 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR12 is enabled. When written, enables AMEVCNTR12.	0b0
[1]	P1	Activity monitor event counter enable bit for AMU.AMEVCNTR11. When N is less than 16, bits [15:N] are **RAZ**, where N is the value in AMU.AMCGCR.CG1NC. Possible values of each bit are: **0b0** When read, means that AMEVCNTR11 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR11 is enabled. When written, enables AMEVCNTR11.	0b0
[0]	P0	Activity monitor event counter enable bit for AMU.AMEVCNTR10. When N is less than 16, bits [15:N] are **RAZ**, where N is the value in AMU.AMCGCR.CG1NC. Possible values of each bit are: **0b0** When read, means that AMEVCNTR10 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR10 is enabled. When written, enables AMEVCNTR10.	0b0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

If there are no auxiliary monitor event counters implemented, reads of AMCNTENSET1 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

There are no implemented auxiliary activity monitor event counters when AMU.AMCFGR.NCG == `0b0000`.

Component	Offset	Instance	Range
AMU	0xC04	AMCNTENSET1	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.19 AMCNTENCLR0, Activity Monitors Count Enable Clear Register 0

Disable control bits for the architected activity monitors event counters, AMU.AMEVCNTR0<n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xC20

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx 0000 0000 0000 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-19: AMU_AMCNTENCLR0 bit assignments

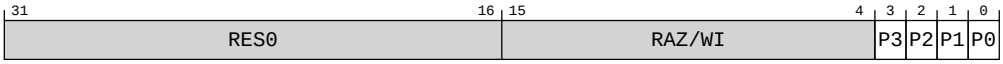


Table B-38: AMCNTENCLR0 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:4]	**RAZ/WI**	Reserved	**RAZ/WI**
[3]	P3	Activity monitor event counter disable bit for AMU.AMEVCNTR03. **Note:** AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4. Possible values of each bit are: **0b0** When read, means that AMEVCNTR03 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR03 is enabled. When written, disables AMEVCNTR03.	0b0
[2]	P2	Activity monitor event counter disable bit for AMU.AMEVCNTR02. **Note:** AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4. Possible values of each bit are: **0b0** When read, means that AMEVCNTR02 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR02 is enabled. When written, disables AMEVCNTR02.	0b0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[1]	P1	Activity monitor event counter disable bit for AMU.AMEVCNTR01. **Note:** AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4. Possible values of each bit are: **0b0** When read, means that AMEVCNTR01 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR01 is enabled. When written, disables AMEVCNTR01.	0b0
[0]	P0	Activity monitor event counter disable bit for AMU.AMEVCNTR00. **Note:** AMU.AMCGCR.CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4. Possible values of each bit are: **0b0** When read, means that AMEVCNTR00 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR00 is enabled. When written, disables AMEVCNTR00.	0b0

Accessibility

Component	Offset	Instance	Range
AMU	0xC20	AMCNTENCLR0	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.20 AMCNTENCLR1, Activity Monitors Count Enable Clear Register 1

Disable control bits for the auxiliary activity monitors event counters, AMU.AMEVCNTR1<n>.

Configurations

This register is available in all configurations.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

32

Component

AMU

Register offset

0xC24

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx 0000 0000 0000 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-20: AMU_AMCNTENCLR1 bit assignments

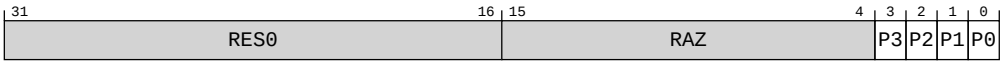


Table B-40: AMCNTENCLR1 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:4]	**RAZ**	Reserved	**RAZ**
[3]	P3	Activity monitor event counter disable bit for AMU.AMEVCNTR13. When N is less than 16, bits [15:N] are **RAZ**, where N is the value in AMU.AMCGCR.CG1NC. Possible values of each bit are: **0b0** When read, means that AMEVCNTR13 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR13 is enabled. When written, disables AMEVCNTR13.	0b0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[2]	P2	Activity monitor event counter disable bit for AMU.AMEVCNTR12. When N is less than 16, bits [15:N] are **RAZ**, where N is the value in AMU.AMCGCR.CG1NC. Possible values of each bit are: **0b0** When read, means that AMEVCNTR12 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR12 is enabled. When written, disables AMEVCNTR12.	0b0
[1]	P1	Activity monitor event counter disable bit for AMU.AMEVCNTR11. When N is less than 16, bits [15:N] are **RAZ**, where N is the value in AMU.AMCGCR.CG1NC. Possible values of each bit are: **0b0** When read, means that AMEVCNTR11 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR11 is enabled. When written, disables AMEVCNTR11.	0b0
[0]	P0	Activity monitor event counter disable bit for AMU.AMEVCNTR10. When N is less than 16, bits [15:N] are **RAZ**, where N is the value in AMU.AMCGCR.CG1NC. Possible values of each bit are: **0b0** When read, means that AMEVCNTR10 is disabled. When written, has no effect. **0b1** When read, means that AMEVCNTR10 is enabled. When written, disables AMEVCNTR10.	0b0

Accessibility

If there are no auxiliary monitor event counters implemented, reads of AMCNTENCLR1 are **RAZ**. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

There are no implemented auxiliary activity monitor event counters when AMU.AMCFGR.NCG == `0b0000`.

Component	Offset	Instance	Range
AMU	0xC24	AMCNTENCLR1	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.21 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

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0100 0000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-21: AMU_AMCGCR bit assignments

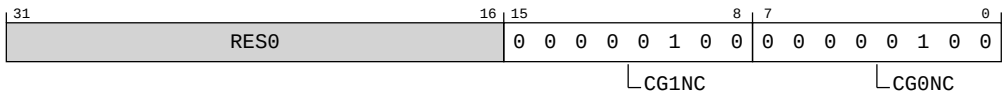


Table B-42: AMCGCR bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:8]	CG1NC	Counter Group 1 Number of Counters. The number of counters in the auxiliary counter group. In an implementation that includes FEAT_AMUv1, the permitted range of values is 0 to 16. **0x04**	0x04

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[7:0]	CG0NC	Counter Group 0 Number of Counters. The number of counters in the architected counter group. **0x04**	0x04

Accessibility

Component	Offset	Instance	Range
AMU	0xCE0	AMCGCR	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.22 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

Access type

RO

Reset value

```
0001 0001 0000 0000 0011 1111 0000 0111
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-22: AMU_AMCFGR bit assignments

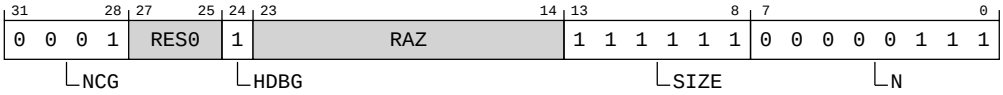


Table B-44: AMCFGR bit descriptions

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	0b0001
[27:25]	**RES0**	Reserved	**RES0**
[24]	HDBG	Halt-on-debug supported. This feature must be supported, and so this bit is 0b1. **0b1** AMU.AMCR.HDBG is read/write.	0b1
[23:14]	**RAZ**	Reserved	**RAZ**
[13:8]	SIZE	Defines the size of the activity monitor event counters, minus one. The counters are 64-bit, 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. The counters are at doubleword-aligned addresses. **0b111111**	0b111111
[7:0]	N	Defines the number of activity monitor event counters implemented in all groups, minus one. **0x07** Eight activity monitor event counters	0x07

Accessibility

Component	Offset	Instance	Range
AMU	0xE00	AMCFGR	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.1.23 AMCR, Activity Monitors Control Register

Global control register for the activity monitors implementation. AMCR 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

0xE04

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-23: AMU_AMCR bit assignments

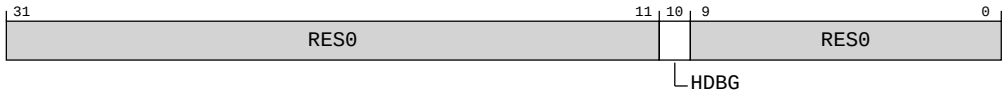


Table B-46: AMCR bit descriptions

Bits	Name	Description	Reset
[31:11]	**RES0**	Reserved	**RES0**
[10]	HDBG	This bit controls whether activity monitor counting is halted when the PE is halted in Debug state. **0b0** Activity monitors do not halt counting when the PE is halted in Debug state. **0b1** Activity monitors halt counting when the PE is halted in Debug state.	x

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[9:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
AMU	0xE04	AMCR	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.24 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

Access type

RO

Reset value

```
1101 1000 1101 0001 0010 0100 0011 1011
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-24: AMU_AMIIDR bit assignments

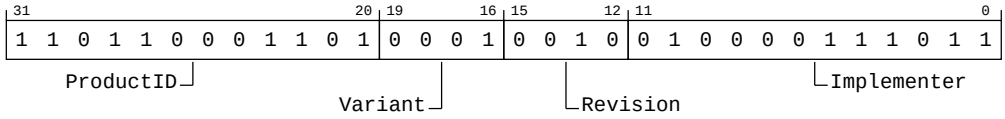


Table B-48: AMIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	This field is an AMU part identifier. **`0xD8D`** C1-SME2	`0xD8D`
[19:16]	Variant	This field distinguishes product variants or major revisions of the product. **`0b0001`** r1p2	`0b0001`
[15:12]	Revision	This field distinguishes minor revisions of the product. **`0b0010`** r1p2	`0b0010`
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the AMU. For an Arm implementation, this field reads as `0x43B`. **`0x43B`** Arm Limited	`0x43B`

Accessibility

Component	Offset	Instance	Range
AMU	0xE08	AMIIDR	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.1.25 AMDEVAFF0, Activity Monitors Device Affinity Register 0

Device Affinity for the AMU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFA8

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-25: AMU_AMDEVAFF0 bit assignments

Table B-50: AMDEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
AMU	0xFA8	AMDEVAFF0	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Otherwise

ERROR

B.1.26 AMDEVAFF1, Activity Monitors Device Affinity Register 1

Device Affinity for the AMU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFAC

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-26: AMU_AMDEVAFF1 bit assignments

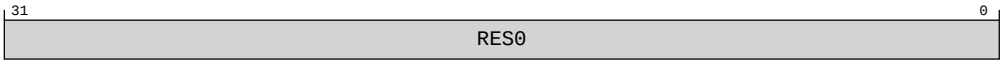


Table B-52: AMDEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
AMU	0xFAC	AMDEVAFF1	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.27 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

Access type

RO

Reset value

```
0100 0111 0111 0000 0000 1010 0110 0110
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-27: AMU_AMDEVARCH bit assignments

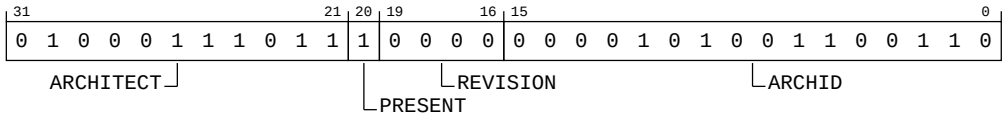


Table B-54: AMDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architect of the component. For Activity Monitors, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, `0b0100`. Bits [27:21] are the JEP106 identification code, `0b0111011`. **`0b01000111011`**	`0b01000111011`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[20]	PRESENT	DEVARCH present. Indicates that the AMDEVARCH register is present. **`0b1`**	`0b1`
[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.	`0b0000`
[15:0]	ARCHID	Defines this part to be an AMU component. For architectures defined by Arm this is further subdivided. For AMU: • Bits [15:12] are the architecture version, also identified as AMDEVARCH.ARCHVER. • Bits [11:0] are the architecture part number, also identified as AMDEVARCH.ARCHPART. AMDEVARCH.ARCHVER = `0x0`, which corresponds to AMU architecture version AMUv1. If FEAT_AMU_EXT32 is implemented, AMDEVARCH is `0xA66`. **`0x0A66`** AMUv1, with FEAT_AMU_EXT32 implemented.	`0x0A66`

Accessibility

Component	Offset	Instance	Range
AMU	0xFBC	AMDEVARCH	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.28 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

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Component

AMU

Register offset

0xFCC

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0001 0110
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-28: AMU_AMDEVTYPE bit assignments

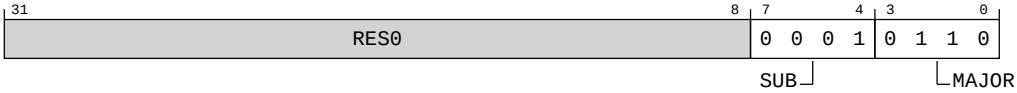


Table B-56: AMDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Subtype. **0b0001** Component within a PE.	0b0001
[3:0]	MAJOR	Major type. **0b0110** Performance monitor component	0b0110

Accessibility

Component	Offset	Instance	Range
AMU	0xFCC	AMDEVTYPE	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.1.29 AMPIDR4, Activity Monitors Peripheral Identification Register 4

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFD0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-29: AMU_AMPIDR4 bit assignments

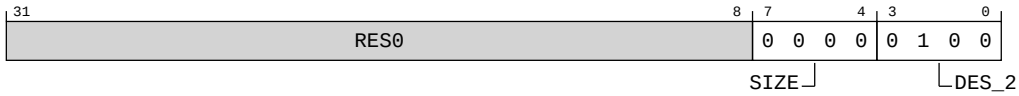


Table B-58: AMPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	Size of the component. Log_2 of the number of 4KB pages from the start of the component to the end of the component ID registers. **0b0000**	0b0000
[3:0]	DES_2	Designer. JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **0b0100** Arm Limited	0b0100

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

Component	Offset	Instance	Range
AMU	0xFD0	AMPIDR4	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.30 AMPIDR0, Activity Monitors Peripheral Identification Register 0

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1000 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-30: AMU_AMPIDR0 bit assignments

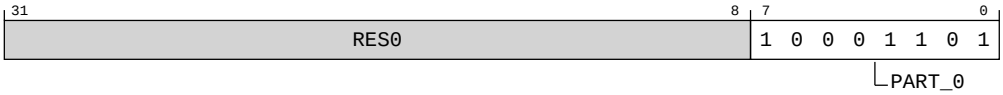


Table B-60: AMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0x8D** C1-SME2	0x8D

Accessibility

Component	Offset	Instance	Range
AMU	0xFE0	AMPIDR0	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.31 AMPIDR1, Activity Monitors Peripheral Identification Register 1

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Register offset

0xFE4

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1011 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-31: AMU_AMPIDR1 bit assignments

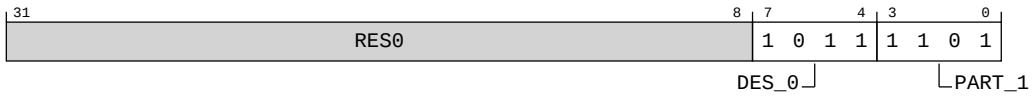


Table B-62: AMPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is `0b1011`. **`0b1011`** Arm Limited	`0b1011`
[3:0]	PART_1	Part number, most significant nibble. **`0b1101`** C1-SME2	`0b1101`

Accessibility

Component	Offset	Instance	Range
AMU	0xFE4	AMPIDR1	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.1.32 AMPIDR2, Activity Monitors Peripheral Identification Register 2

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE8

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0001 1011
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-32: AMU_AMPIDR2 bit assignments

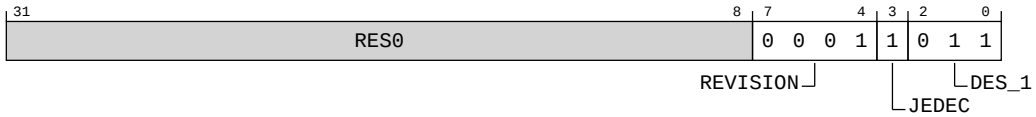


Table B-64: AMPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0001** r1p2	0b0001
[3]	JEDEC	Indicates a JEP106 identity code is used. **0b1**	0b1

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is `0b011`. **`0b011`** Arm Limited	`0b011`

Accessibility

Component	Offset	Instance	Range
AMU	0xFE8	AMPIDR2	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.33 AMPIDR3, Activity Monitors Peripheral Identification Register 3

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFEC

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0010 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-33: AMU_AMPIDR3 bit assignments

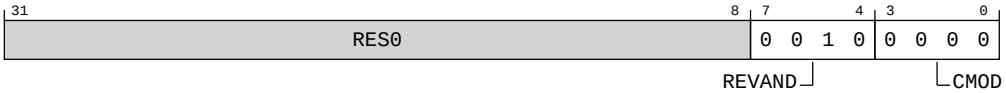


Table B-66: AMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Part minor revision. Parts using AMU.AMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0010** r1p2	0b0010
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
AMU	0xFEC	AMPIDR3	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.34 AMCIDR0, Activity Monitors Component Identification Register 0

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Component

AMU

Register offset

0xFF0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-34: AMU_AMCIDR0 bit assignments

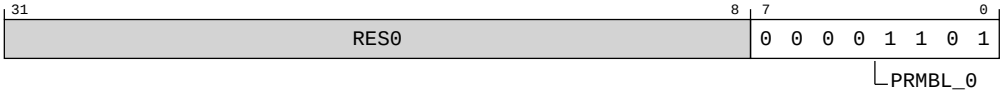


Table B-68: AMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble. **0x0D**	0x0D

Accessibility

Component	Offset	Instance	Range
AMU	0xFF0	AMCIDR0	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.1.35 AMCIDR1, Activity Monitors Component Identification Register 1

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFF4

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1001 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-35: AMU_AMCIDR1 bit assignments

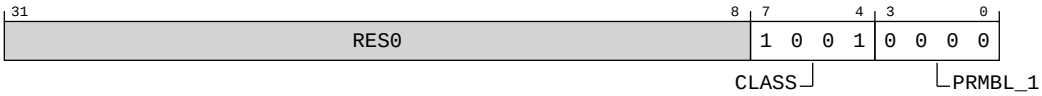


Table B-70: AMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight component. Other values are defined by the CoreSight Architecture. This field reads as `0x9`.	`0b1001`
[3:0]	PRMBL_1	Preamble. **0b0000**	`0b0000`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

Component	Offset	Instance	Range
AMU	0xFF4	AMCIDR1	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.36 AMCIDR2, Activity Monitors Component Identification Register 2

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFF8

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-36: AMU_AMCIDR2 bit assignments

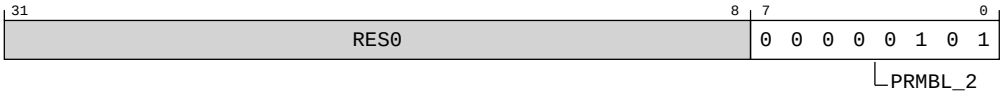


Table B-72: AMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble. **0x05**	0x05

Accessibility

Component	Offset	Instance	Range
AMU	0xFF8	AMCIDR2	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.1.37 AMCIDR3, Activity Monitors Component Identification Register 3

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Register offset

0xFFC

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1011 0001
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-37: AMU_AMCIDR3 bit assignments

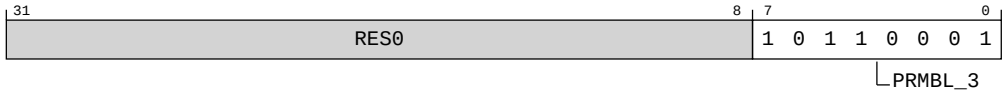


Table B-74: AMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble. **0xB1**	0xB1

Accessibility

Component	Offset	Instance	Range
AMU	0xFFC	AMCIDR3	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.2 External MPMM registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped MPMM registers in the core.

For more information on registers listed in the table, click on the link associated with the register name.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the *Arm® Architecture Reference Manual for A-profile architecture*.

Table B-76: MPMM registers summary

Offset	Name	Reset	Width	Description
0x000	CMEPPMCR	See individual bit resets.	64-bit	Global PPM Configuration Register
0x010	CMEMPMMCR [31:0]	See individual bit resets.	32-bit	Global MPMM Configuration Register
0x014	CMEMPMMCR [63:32]	See individual bit resets.	32-bit	Global MPMM Configuration Register
0x090	CMEMPMMTUNE [31:0]	See individual bit resets.	32-bit	MPMM Tuning Configuration Register
0x094	CMEMPMMTUNE [63:32]	See individual bit resets.	32-bit	MPMM Tuning Configuration Register

B.2.1 CMEPPMCR, Global PPM Configuration Register

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

Configurations

External register CMEPPMCR bits [31:0] are architecturally mapped to AArch64 System register A.4.1 IMP_CMEPPMCR_EL3, Global PPM Configuration Register on page 161 bits [31:0].

Attributes

Width

64

Component

MPMM

Register offset

0x000

Access type

RW

Reset value

```
0xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x011 xxxx xxx0
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-38: EXT_CMEPPMCR bit assignments

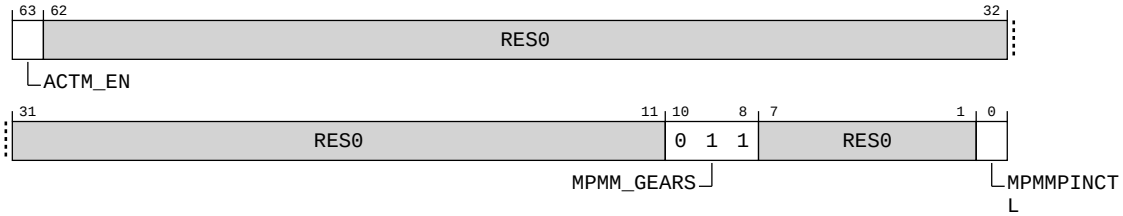


Table B-77: CMEPPMCR bit descriptions

Bits	Name	Description	Reset
[63]	ACTM_EN	Activity Meter Enable **0b0** Disable Activity Meter logic pins. **0b1** Enable Activity Meter logic pins.	0b0
[62:11]	**RES0**	Reserved	**RES0**
[10:8]	MPMM_GEARS	Number of MPMM Gears implemented **0b011** 3 MPMM gears are available.	0b011
[7:1]	**RES0**	Reserved	**RES0**
[0]	MPMMPINCTL	MPMM Pin Control Enabled **0b0** MPMM control through SPR and utility bus. **0b1** MPMM control through pin only.	0b0

Accessibility

Component	Offset	Instance	Range
MPMM	0x000	CMEPPMCR	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.2.2 CMEMPMMCR, Global MPMM Configuration Register

This register is used to change MPMM gears or disable MPMM.

Configurations

External register CMEMPMMCR bits [63:0] are architecturally mapped to AArch64 System register A.2.2 IMP_CMEMPMMCR_EL3, Global MPMM Configuration Register on page 123 bits [63:0].

Attributes

Width

64

Component

MPMM

Register offsets (2)

0x010,0x014

Access type

RW

Reset value

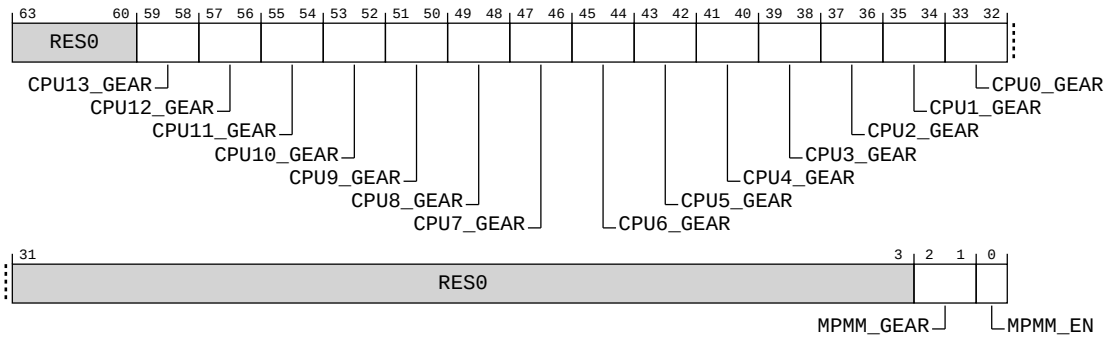
```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx11 xxxx xxxx xxxx xxxx xxxx xxxx xxxx x000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-39: EXT_CMEMPMMCR bit assignments



Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table B-79: CMEMPMMCR bit descriptions

Bits	Name	Description	Reset
[63:60]	**RES0**	Reserved	**RES0**
[59:58]	CPU13_GEAR	**When NUM_CORES == 14** Gear for CPU13. The most constraining gear between CPU13_GEAR and MPMM_GEAR is applied to CPU13. **0b00** Maximum selected gear for CPU13 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU13 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU13 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU13 is MPMM_GEAR. **Otherwise** RES0	`0b11`
[57:56]	CPU12_GEAR	**When NUM_CORES >= 13** Gear for CPU12. The most constraining gear between CPU12_GEAR and MPMM_GEAR is applied to CPU12. **0b00** Maximum selected gear for CPU12 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU12 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU12 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU12 is MPMM_GEAR. **Otherwise** RES0	`0b11`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[55:54]	CPU11_GEAR	**When NUM_CORES >= 12** Gear for CPU11. The most constraining gear between CPU11_GEAR and MPMM_GEAR is applied to CPU11. **0b00** Maximum selected gear for CPU11 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU11 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU11 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU11 is MPMM_GEAR. **Otherwise** RES0	0b11
[53:52]	CPU10_GEAR	**When NUM_CORES >= 11** Gear for CPU10. The most constraining gear between CPU10_GEAR and MPMM_GEAR is applied to CPU10. **0b00** Maximum selected gear for CPU10 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU10 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU10 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU10 is MPMM_GEAR. **Otherwise** RES0	0b11
[51:50]	CPU9_GEAR	**When NUM_CORES >= 10** Gear for CPU9. The most constraining gear between CPU9_GEAR and MPMM_GEAR is applied to CPU9. **0b00** Maximum selected gear for CPU9 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU9 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU9 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU9 is MPMM_GEAR. **Otherwise** RES0	0b11

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[49:48]	CPU8_GEAR	**When NUM_CORES >= 9** Gear for CPU8. The most constraining gear between CPU8_GEAR and MPMM_GEAR is applied to CPU8. **0b00** Maximum selected gear for CPU8 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU8 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU8 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU8 is MPMM_GEAR. **Otherwise** RES0	0b11
[47:46]	CPU7_GEAR	**When NUM_CORES >= 8** Gear for CPU7. The most constraining gear between CPU7_GEAR and MPMM_GEAR is applied to CPU7. **0b00** Maximum selected gear for CPU7 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU7 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU7 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU7 is MPMM_GEAR. **Otherwise** RES0	0b11
[45:44]	CPU6_GEAR	**When NUM_CORES >= 7** Gear for CPU6. The most constraining gear between CPU6_GEAR and MPMM_GEAR is applied to CPU6. **0b00** Maximum selected gear for CPU6 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU6 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU6 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU6 is MPMM_GEAR. **Otherwise** RES0	0b11

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[43:42]	CPU5_GEAR	**When NUM_CORES >= 6** Gear for CPU5. The most constraining gear between CPU5_GEAR and MPMM_GEAR is applied to CPU5. **0b00** Maximum selected gear for CPU5 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU5 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU5 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU5 is MPMM_GEAR. **Otherwise** RES0	0b11
[41:40]	CPU4_GEAR	**When NUM_CORES >= 5** Gear for CPU4. The most constraining gear between CPU4_GEAR and MPMM_GEAR is applied to CPU4. **0b00** Maximum selected gear for CPU4 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU4 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU4 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU4 is MPMM_GEAR. **Otherwise** RES0	0b11
[39:38]	CPU3_GEAR	**When NUM_CORES >= 4** Gear for CPU3. The most constraining gear between CPU3_GEAR and MPMM_GEAR is applied to CPU3. **0b00** Maximum selected gear for CPU3 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU3 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU3 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU3 is MPMM_GEAR. **Otherwise** RES0	0b11

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[37:36]	CPU2_GEAR	**When NUM_CORES >= 3** Gear for CPU2. The most constraining gear between CPU2_GEAR and MPMM_GEAR is applied to CPU2. **0b00** Maximum selected gear for CPU2 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU2 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU2 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU2 is MPMM_GEAR. **Otherwise** RES0	0b11
[35:34]	CPU1_GEAR	**When NUM_CORES >= 2** Gear for CPU1. The most constraining gear between CPU1_GEAR and MPMM_GEAR is applied to CPU1. **0b00** Maximum selected gear for CPU1 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU1 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU1 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU1 is MPMM_GEAR. **Otherwise** RES0	0b11
[33:32]	CPU0_GEAR	Gear for CPU0. The most constraining gear between CPU0_GEAR and MPMM_GEAR is applied to CPU0 **0b00** Maximum selected gear for CPU0 is Gear 0 even if MPMM_GEAR is bigger. **0b01** Maximum selected gear for CPU0 is Gear 1 even if MPMM_GEAR is bigger. **0b10** Maximum selected gear for CPU0 is Gear 2 even if MPMM_GEAR is bigger. **0b11** Maximum selected gear for CPU0 is MPMM_GEAR.	0b11
[31:3]	**RES0**	Reserved	**RES0**

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[2:1]	MPMM_GEAR	MPMM Gear Select **0b00** Select MPMM Gear 0. **0b01** Select MPMM Gear 1. **0b10** Select MPMM Gear 2. **0b11** Do not throttle.	0b00
[0]	MPMM_EN	MPMM Enable **0b0** MPMM is not enabled. **0b1** MPMM is enabled.	0b0

Accessibility

Component	Offset	Instance	Range
MPMM	0x010	CMEMPMMCR	31:0

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Component	Offset	Instance	Range
MPMM	0x014	CMEMPMMCR	63:32

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.2.3 CMEMPMMTUNE, MPMM Tuning Configuration Register

This register contains the fine tuning/trim configuration for MPMM.

Configurations

External register CMEMPMMTUNE bits [63:0] are architecturally mapped to AArch64 System register A.4.2 IMP_CMEMPMMTUNE_EL3, MPMM Tuning Configuration Register on page 164 bits [63:0].

Attributes

Width

64

Component

MPMM

Register offsets (2)

0x090,0x094

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

This register contains control bits that affect the CPU behavior

Figure B-40: EXT_CMEMPMMTUNE bit assignments

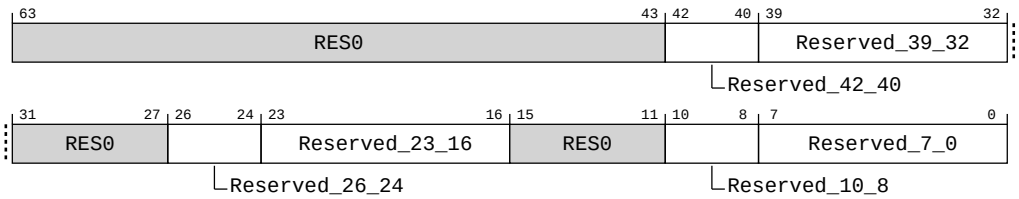


Table B-82: CMEMPMMTUNE bit descriptions

Bits	Name	Description	Reset
[63:43]	RES0	Reserved	RES0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[42:40]	Reserved_42_40	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_42_40 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_42_40 **Otherwise** RES0	`0b001`
[39:32]	Reserved_39_32	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_39_32 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_39_32 **Otherwise** RES0	The reset values can be the following: `0x83`, `0x85`, respective to the value
[31:27]	**RES0**	Reserved	**RES0**
[26:24]	Reserved_26_24	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_26_24 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_26_24 **Otherwise** RES0	`0b010`
[23:16]	Reserved_23_16	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_23_16 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_23_16 **Otherwise** RES0	The reset values can be the following: `0x6B`, `0x70`, respective to the value
[15:11]	**RES0**	Reserved	**RES0**
[10:8]	Reserved_10_8	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_10_8 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_10_8 **Otherwise** RES0	`0b011`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[7:0]	Reserved_7_0	**When IsFeatureImplemented(FULL_MATMUL)** Reserved_7_0 **When ! IsFeatureImplemented(FULL_MATMUL)** Reserved_7_0 **Otherwise** RES0	0x50

Accessibility

Component	Offset	Instance	Range
MPMM	0x090	CMEMPMMTUNE	31:0

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()
RW

When IsCorePowered() and !IsAccessSecure()
RAZ/WI

Otherwise
ERROR

Component	Offset	Instance	Range
MPMM	0x094	CMEMPMMTUNE	63:32

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()
RW

When IsCorePowered() and !IsAccessSecure()
RAZ/WI

Otherwise
ERROR

B.3 External RAS registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped RAS registers in the core.

For more information on registers listed in the table, click on the link associated with the register name.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the *Arm® Architecture Reference Manual for A-profile architecture*.

Table B-85: RAS registers summary

Offset	Name	Reset	Width	Description
0x0	ERR0FR	`0x005100008010A9A2`	64-bit	Error Record <n> Feature Register
0x8	ERR0CTLR	See individual bit resets.	64-bit	Error Record <n> Control Register
0x10	ERR0STATUS	See individual bit resets.	64-bit	Error Record <n> Primary Status Register
0x18	ERR0ADDR	See individual bit resets.	64-bit	Error Record <n> Address Register
0x20	ERR0MISC0	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 0
0x28	ERR0MISC1	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 1
0x30	ERR0MISC2	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 2
0x38	ERR0MISC3	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 3
0x800	ERR0PFGF	`0x0000000070001A63`	64-bit	Error Record <n> Pseudo-fault Generation Feature Register
0x808	ERR0PFGCTL	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Control Register
0x810	ERR0PFGCDN	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Countdown Register
0xE00	ERRGSR	See individual bit resets.	64-bit	Error Group Status Register
0xE10	ERRIIDR	`0xD8D1243B`	32-bit	Implementation Identification Register
0xFA8	ERRDEVAFF	See individual bit resets.	64-bit	Device Affinity Register
0xFBC	ERRDEVARCH	`0x47710A00`	32-bit	Device Architecture Register
0xFC8	ERRDEVID	`0x00000001`	32-bit	Device Configuration Register
0xFD0	ERRPIDR4	`0x00000004`	32-bit	Peripheral Identification Register 4
0xFE0	ERRPIDR0	`0x0000008D`	32-bit	Peripheral Identification Register 0
0xFE4	ERRPIDR1	`0x000000BD`	32-bit	Peripheral Identification Register 1
0xFE8	ERRPIDR2	`0x0000001B`	32-bit	Peripheral Identification Register 2
0xFEC	ERRPIDR3	`0x00000020`	32-bit	Peripheral Identification Register 3
0xFF0	ERRCIDR0	`0x0000000D`	32-bit	Component Identification Register 0
0xFF4	ERRCIDR1	`0x000000F0`	32-bit	Component Identification Register 1
0xFF8	ERRCIDR2	`0x00000005`	32-bit	Component Identification Register 2
0xFFC	ERRCIDR3	`0x000000B1`	32-bit	Component Identification Register 3

B.3.1 ERR0FR, Error Record <n> Feature Register

Defines whether error record 0 is the first record owned by a node:

- If error record 0 is the first error record owned by a node, then ERR0FR.ED is not `0b00`.
- If error record 0 is not the first error record owned by a node, then ERR0FR.ED is `0b00`.

If error record 0 is the first record owned by the node, defines which of the common architecturally-defined features are implemented by the node and, of the implemented features, which are software programmable.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

RAS

Register offset

0x0

Access type

RO

Reset value

```
0000 0000 0101 0001 0000 0000 0000 0000 1000 0000 0001 0000 1010 1001 1010 0010
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-41: EXT_ERR0FR bit assignments

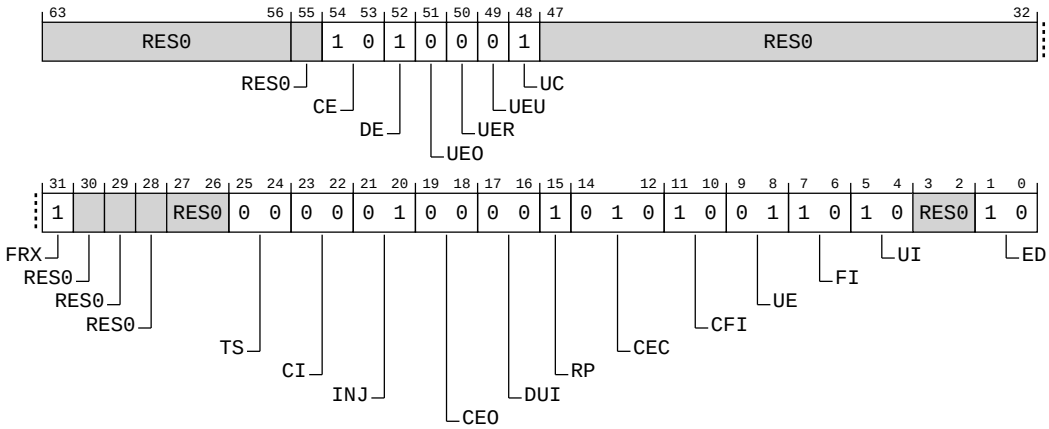


Table B-86: ERR0FR bit descriptions

Bits	Name	Description	Reset
[63:55]	**RES0**	Reserved	**RES0**
[54:53]	CE	Corrected Error recording. Describes the types of Corrected errors the node can record, if any. **0b10** Records only non-specific Corrected errors. That is, Corrected errors recorded by setting ERR0STATUS.CE to `0b10`.	`0b10`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[52]	DE	Deferred Error recording. Describes whether the node supports recording Deferred errors. **0b1** Records Deferred errors.	0b1
[51]	UEO	Latent or Restartable Error recording. Describes whether the node supports recording Latent or Restartable errors. **0b0** Does not record Latent or Restartable errors.	0b0
[50]	UER	Signaled or Recoverable Error recording. Describes whether the node supports recording Signaled or Recoverable errors. **0b0** Does not record Signaled or Recoverable errors.	0b0
[49]	UEU	Unrecoverable Error recording. Describes whether the node supports recording Unrecoverable errors. **0b0** Does not record Unrecoverable errors.	0b0
[48]	UC	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. **0b1** Records Uncontainable errors.	0b1
[47:32]	**RES0**	Reserved	**RES0**
[31]	FRX	Feature Register extension. Defines whether ERR0FR[63:48] are architecturally defined. **0b1** ERR0FR[63:48] are defined by the architecture.	0b1
[30:26]	**RES0**	Reserved	**RES0**
[25:24]	TS	Timestamp Extension. Indicates whether, for each error record <m> owned by this node, ERR<m>MISC3 is used as the timestamp register, and, if it is, the timebase used by the timestamp. **0b00** Does not support a timestamp register. All other values are reserved.	0b00
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented by the node. **0b00** Does not support the critical error interrupt. ERR0CTLR.CI is **RES0**. All other values are reserved.	0b00
[21:20]	INJ	Fault Injection Extension. Indicates whether the Common Fault Injection Model Extension is implemented by the node. **0b01** Supports the Common Fault Injection Model Extension. See ERR0PFGF for more information. All other values are reserved.	0b01

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior of the node when a second or subsequent Corrected error is recorded and a first Corrected error has previously been recorded by an error record <m> owned by the node. **0b00** Keeps the previous error syndrome. All other values are reserved. The second or subsequent Corrected error is counted by the Corrected error counter, regardless of the value of this field. If counting the error causes unsigned overflow of the counter, then ERR<m>STATUS.OF is set to 1. This means that, if no other error is subsequently recorded that overwrites the syndrome: • If ERR0FR.CEO is `0b00`, the error record holds the syndrome for the first recorded Corrected error. • If ERR0FR.CEO is `0b01`, the error record holds the syndrome for the most recently recorded Corrected error before the counter overflows.	`0b00`
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the enabling and disabling of error recovery interrupts on deferred errors is supported by the node. **0b00** Does not support the enabling and disabling of error recovery interrupts on deferred errors. ERR0CTLR.DUI is **RES0**. All other values are reserved.	`0b00`
[15]	RP	Repeat counter. Indicates whether the node implements a second Corrected error counter in ERR<m>MISC0 for each error record <m> owned by the node that can record countable errors. **0b1** Implements a first (repeat) counter and a second (other) counter in ERR<m>MISC0 for each error record <m> owned by the node that can record countable errors. The repeat counter is the same size as the primary error counter.	`0b1`
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter mechanisms in ERR<m>MISC0 for each error record <m> owned by the node that can record countable errors. **0b010** Implements an 8-bit Corrected error counter in ERR<m>MISC0[39:32] for each error record <m> owned by the node that can record countable errors. All other values are reserved. **Note:** Implementations might include other error counter models, or might include the standard model and not indicate this in ERR0FR.	`0b010`
[11:10]	CFI	Fault handling interrupt for corrected errors control. Indicates whether the enabling and disabling of fault handling interrupts on corrected errors is supported by the node. **0b10** Enabling and disabling of fault handling interrupts on corrected errors is supported and controllable using ERR0CTLR.CFI. All other values are reserved.	`0b10`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[9:8]	UE	In-band error reponse (External Abort). Indicates whether the in-band error response and associated controls are implemented by the node. **0b01** In-band error response is supported and always enabled. ERR0CTLR.UE is **RES0**.	0b01
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented by the node. **0b10** Fault handling interrupt is supported and controllable using ERR0CTLR.FI.	0b10
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented by the node. **0b10** Error handling interrupt is supported and controllable using ERR0CTLR.UI.	0b10
[3:2]	**RES0**	Reserved	**RES0**
[1:0]	ED	Error reporting and logging. Indicates error record <n> is the first record owned the node, and whether the node implements the controls for enabling and disabling error reporting and logging. **0b10** Error reporting and logging is controllable using ERR0CTLR.ED. All other values are reserved.	0b10

Accessibility

Component	Offset	Instance	Range
RAS	0x0	ERR0FR	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.2 ERR0CTLR, Error Record <n> Control Register

The error control register contains enable bits for the node that writes to this record:

- Enabling error detection and correction.
- Enabling the critical error, error recovery, and fault handling interrupts.
- Enabling in-band error response for uncorrected errors.

For each bit, if the node does not support the feature, then the bit is **RES0**. The definition of each record is **IMPLEMENTATION DEFINED**.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Configurations

ERR0FR contains additional information about the node.

Attributes

Width

64

Component

RAS

Register offset

0x8

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 xxxx 00x0
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-42: EXT_ERR0CTLR bit assignments

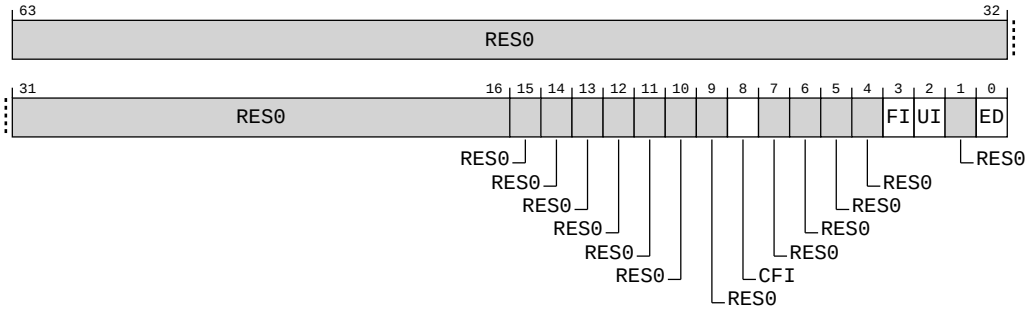


Table B-88: ERR0CTLR bit descriptions

Bits	Name	Description	Reset
[63:9]	RES0	Reserved	RES0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[8]	CFI	Fault handling interrupt for Corrected errors enable. When ERR0FR.CFI == `0b10`, this control applies to errors arising from both reads and writes. When enabled: • If the node implements Corrected error counters, then the fault handling interrupt is generated when a counter overflows and the overflow bit for the counter is set to 1. For more information, see ERR0MISC0. • Otherwise, the fault handling interrupt is also generated for all errors recorded as Corrected error. **`0b0`** Fault handling interrupt not generated for Corrected errors. **`0b1`** Fault handling interrupt generated for Corrected errors. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	`0b0`
[7:4]	**RES0**	Reserved	**RES0**
[3]	FI	Fault handling interrupt enable. When ERR0FR.FI == `0b10`, this control applies to errors arising from both reads and writes. When enabled: • The fault handling interrupt is generated for all errors recorded as either Deferred error or Uncorrected error. • If the fault handling interrupt for Corrected errors control is not implemented: ◦ If the node implements Corrected error counters, then the fault handling interrupt is also generated when a counter overflows and the overflow bit for the counter is set to 1. ◦ Otherwise, the fault handling interrupt is also generated for all errors recorded as Corrected error. **`0b0`** Fault handling interrupt disabled. **`0b1`** Fault handling interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	`0b0`
[2]	UI	Uncorrected error recovery interrupt enable. When ERR0FR.UI == `0b10`, this control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all errors recorded as Uncorrected error. **`0b0`** Error recovery interrupt disabled. **`0b1`** Error recovery interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	`0b0`
[1]	**RES0**	Reserved	**RES0**

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[0]	ED	Error reporting and logging enable. When disabled, the node behaves as if error detection and correction are disabled, and no errors are recorded or signaled by the node. **0b0** Error reporting disabled. **0b1** Error reporting enabled.	0b0

Accessibility

Component	Offset	Instance	Range
RAS	0x8	ERR0CTLR	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.3 ERR0STATUS, Error Record <n> Primary Status Register

Contains status information for error record 0, including:

- Whether any error has been detected (valid).
- Whether any detected error was not corrected, and returned to a Requester.
- Whether any detected error was not corrected and deferred.
- Whether an error record has been discarded because additional errors have been detected before the first error was handled by software (overflow).
- Whether any error has been reported.
- Whether the other error record registers contain valid information.
- Whether the error was reported because poison data was detected or because a corrupt value was detected by an error detection code.
- A primary error code.
- An **IMPLEMENTATION DEFINED** extended error code.

Within this register:

- ERR0STATUS.{AV, V, MV} are valid bits that define whether error record 0 registers are valid.
- ERR0STATUS.{UE, OF, CE, DE, UET} encode the types of error or errors recorded.
- ERR0STATUS.{CI, ER, PN, IERR, SERR} are syndrome fields.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Configurations

ERRFRPFGF[FirstRecordOfNode(n)] describes the features implemented by the node that owns error record 0. FirstRecordOfNode(n) is the index of the first error record owned by the same node as error record 0. If the node owns a single record then FirstRecordOfNode(n) = n.

For **IMPLEMENTATION DEFINED** fields in ERR0STATUS, writing zero returns the error record to an initial quiescent state.

In particular, if any **IMPLEMENTATION DEFINED** syndrome fields might generate a Fault Handling or Error Recovery Interrupt request, writing zero is sufficient to deactivate the Interrupt request.

Fields that are read-only, nonzero, and ignore writes are compliant with this requirement.

Note

Arm recommends that any **IMPLEMENTATION DEFINED** syndrome field that can generate a Fault Handling, Error Recovery, Critical, or **IMPLEMENTATION DEFINED**, interrupt request is disabled at Cold reset and is enabled by software writing an **IMPLEMENTATION DEFINED** nonzero value to an **IMPLEMENTATION DEFINED** field in ERRCTLR[FirstRecordOfNode(n)].

Attributes

Width

64

Component

RAS

Register offset

0x10

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 00xx x0xx xxxx xxxx xxxx xxxx xxx0 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-43: EXT_ERR0STATUS bit assignments

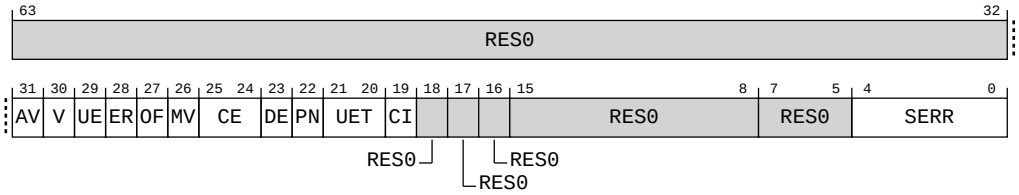


Table B-90: ERR0STATUS bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	AV	Address Valid. **0b0** ERR0ADDR not valid. **0b1** ERR0ADDR contains an address associated with the highest priority error recorded by this record. Access to this field is: W1C	0b0
[30]	V	Status Register Valid. **0b0** ERR0STATUS not valid. **0b1** ERR0STATUS valid. At least one error has been recorded. Access to this field is: W1C	0b0
[29]	UE	Uncorrected Error. **0b0** No errors have been detected, or all detected errors have been either corrected or deferred. **0b1** At least one detected error was not corrected and not deferred. When clearing ERR0STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[28]	ER	Error Reported. **0b0** No in-band error response (External abort) signaled to the Requester making the access or other transaction. **0b1** An in-band error response was signaled by the component to the Requester making the access or other transaction. This can be because any of the following are true: • The ERRCTLR[FirstRecordOfNode(n)].UE field, or applicable one of the ERRCTLR[FirstRecordOfNode(n)].{WUE, RUE} fields, is implemented and was 1 when an error was detected and not corrected. • The ERRCTLR[FirstRecordOfNode(n)].{WUE, RUE, UE} fields are not implemented and the component always reports errors. **Note:** An in-band error response signaled by the component might be masked and not generate any exception. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0' or ext-ERR0STATUS.[DE,UE] == '00'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[27]	OF	Overflow. Indicates that multiple errors have been detected. This field is set to 1 when one of the following occurs: • A Corrected error counter is implemented, an error is counted, and the counter overflows. • ERR0STATUS.V was previously 1, a Corrected error counter is not implemented, and a Corrected error is recorded. • ERR0STATUS.V was previously 1, and a type of error other than a Corrected error is recorded. Otherwise, this field is unchanged when an error is recorded. If a Corrected error counter is implemented, then: • A direct write that modifies the counter overflow flag indirectly might set this field to an **UNKNOWN** value. • A direct write to this field that clears this field to zero might indirectly set the counter overflow flag to an **UNKNOWN** value. **0b0** Since this field was last cleared to zero, no error syndrome has been discarded and, if a Corrected error counter is implemented, it has not overflowed. **0b1** Since this field was last cleared to zero, at least one error syndrome has been discarded or, if a Corrected error counter is implemented, it might have overflowed. When clearing ERR0STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x
[26]	MV	Miscellaneous Registers Valid. **0b0** ERR0MISC<m> not valid. **0b1** The contents of the ERR0MISC<m> registers contains additional information for an error recorded by this record. Access to this field is: W1C	0b0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[25:24]	CE	Corrected Error. **0b00** No errors were corrected. **0b10** At least one error was corrected. When clearing ERR0STATUS.V to 0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	xx
[23]	DE	Deferred Error. **0b0** No errors were deferred. **0b1** At least one error was not corrected and deferred. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x
[22]	PN	Poison. **0b0** Uncorrected error or Deferred error recorded because a corrupt value was detected, for example, by an error detection code (EDC), or Corrected error recorded. **0b1** Uncorrected error or Deferred error recorded because a poison value was detected. When clearing ERR0STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0' or ext-ERR0STATUS.[DE,UE] == '00'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[21:20]	UET	Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. **0b00** Uncorrected error, Uncontainable error (UC). **0b01** Uncorrected error, Unrecoverable error (UEU). **0b10** Uncorrected error, Latent or Restartable error (UEO). **0b11** Uncorrected error, Signaled or Recoverable error (UER). UER can mean either Signaled or Recoverable error, and UEO can mean either Latent or Restartable error. When clearing ERR0STATUS.V to 0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0' or ext-ERR0STATUS.UE == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	xx
[19]	CI	Critical Error. Indicates whether a critical error condition has been recorded. **0b0** No critical error condition. **0b1** Critical error condition. When clearing ERR0STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x
[18:5]	**RES0**	Reserved	**RES0**

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[4:0]	SERR	Architecturally-defined primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry. **0b00000** No error. **0b00010** Data value from (non-associative) internal memory. For example, ECC from on-chip SRAM or buffer. **0b00110** Data value from associative memory. For example, ECC error on cache data. **0b00111** Address/control value from associative memory. For example, ECC error on cache tag. All other values are reserved. This field is not valid and reads **UNKNOWN** if ERR0STATUS.V == `0b0`.	`0b00000`

Accessibility

ERR0STATUS.{AV, V, UE, ER, OF, MV, CE, DE, PN, UET, CI} are write-one-to-clear (W1C) fields, meaning writes of zero are ignored, and a write of one or all-ones to the field clears the field to zero. ERR0STATUS.{IERR, SERR} are read/write (RW) fields, although the set of implemented valid values is **IMPLEMENTATION DEFINED**. See also ERR0PFGF.SYN.

After reading ERR0STATUS, software must clear the valid fields in the register to allow new errors to be recorded. However, between reading the register and clearing the valid fields, a new error might have overwritten the register. To prevent this error being lost by software, the register prevents updates to fields that might have been updated by a new error.

When RAS System Architecture v1.0 is implemented:

- Writes to ERR0STATUS.{UE, DE, CE} are ignored if ERR0STATUS.OF is 1 and is not being cleared to 0.
- Writes to ERR0STATUS.V are ignored if any of ERR0STATUS.{UE, DE, CE} are nonzero and are not being cleared to zero.
- Writes to ERR0STATUS.{AV, MV} and the ERR0STATUS.{ER, PN, UET, IERR, SERR} syndrome fields are ignored if the highest priority nonzero error status field is not being cleared to zero. The error status fields in priority order from highest to lowest, are ERR0STATUS.UE, ERR0STATUS.DE, and ERR0STATUS.CE.

When RAS System Architecture v1.1 is implemented, a write to the register is ignored if all of:

- Any of ERR0STATUS.{V, UE, OF, CE, DE} are nonzero before the write.
- The write does not clear the nonzero ERR0STATUS.{V, UE, OF, CE, DE} fields to zero by writing ones to the applicable field or fields.

Some of the fields in ERR0STATUS are also defined as **UNKNOWN** where certain combinations of ERR0STATUS.{V, DE, UE} are zero. The rules for writes to ERR0STATUS allow a node to implement such a field as a fixed read-only value.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

For example, when RAS System Architecture v1.1 is implemented, a write to ERR0STATUS when ERR0STATUS.V is 1 results in either ERR0STATUS.V field being cleared to zero, or ERR0STATUS.V not changing. Since all fields in ERR0STATUS, other than ERR0STATUS.{AV, V, MV}, usually read as **UNKNOWN** values when ERR0STATUS.V is zero, this means those fields can be implemented as read-only if applicable.

To ensure correct and portable operation, when software is clearing the valid fields in the register to allow new errors to be recorded, Arm recommends that software performs the following sequence of operations in order:

1. Read ERR0STATUS and determine which fields need to be cleared to zero.
2. In a single write to ERR0STATUS:
   - Write ones to all the W1C fields that are nonzero in the read value.
   - Write zero to all the W1C fields that are zero in the read value.
   - Write zero to all the RW fields.
3. Read back ERR0STATUS after the write to confirm no new fault has been recorded.

Otherwise, these fields might not have the correct value when a new fault is recorded.

Component	Offset	Instance	Range
RAS	0x10	ERR0STATUS	None

This interface is accessible as follows:

When IsCorePowered(), IsAccessSecure(), ext-ERR0STATUS.V != '0' and ERR0STATUS.V is not being cleared to 0b0 in the same write

RO

When IsCorePowered(), IsAccessSecure(), ext-ERR0STATUS.UE != '0' and ERR0STATUS.UE is not being cleared to 0b0 in the same write

RO

When IsCorePowered(), IsAccessSecure(), ext-ERR0STATUS.OF != '0' and ERR0STATUS.OF is not being cleared to 0b0 in the same write

RO

When IsCorePowered(), IsAccessSecure(), ext-ERR0STATUS.CE != '00' and ERR0STATUS.CE is not being cleared to 0b00 in the same write

RO

When IsCorePowered(), IsAccessSecure(), ext-ERR0STATUS.DE != '0' and ERR0STATUS.DE is not being cleared to 0b0 in the same write

RO

When IsCorePowered(), IsAccessSecure() and (ext-ERR0STATUS.V == '0' or ERR0STATUS.V is being cleared to 0b0 in the same write)

RW

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When IsCorePowered(), IsAccessSecure() and (ext-ERR0STATUS.UE == '0' or ERR0STATUS.UE is being cleared to 0b0 in the same write)

RW

When IsCorePowered(), IsAccessSecure() and (ext-ERR0STATUS.OF == '0' or ERR0STATUS.OF is being cleared to 0b0 in the same write)

RW

When IsCorePowered(), IsAccessSecure() and (ext-ERR0STATUS.CE == '00' or ERR0STATUS.CE is being cleared to 0b00 in the same write)

RW

When IsCorePowered(), IsAccessSecure() and (ext-ERR0STATUS.DE == '0' or ERR0STATUS.DE is being cleared to 0b0 in the same write)

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.4 ERR0ADDR, Error Record <n> Address Register

If an address is associated with a detected error, then it is written to ERR0ADDR when the error is recorded

Configurations

ERRFR[FirstRecordOfNode(n)] describes the features implemented by the node that owns error record 0. FirstRecordOfNode(n) is the index of the first error record owned by the same node as error record 0. If the node owns a single record then FirstRecordOfNode(n) = n.

Attributes

Width

64

Component

RAS

Register offset

0x18

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-44: EXT_ERR0ADDR bit assignments

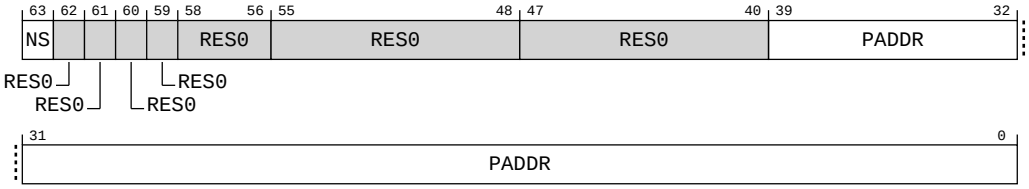


Table B-92: ERR0ADDR bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure attribute. **0b0** The address is Secure. **0b1** The address is Non-secure.	x
[62:40]	**RES0**	Reserved	**RES0**
[39:0]	PADDR	Physical Address. Address of the recorded location.	40{x}

Accessibility

Component	Offset	Instance	Range
RAS	0x18	ERR0ADDR	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.3.5 ERR0MISC0, Error Record <n> Miscellaneous Register 0

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to locate where the error was detected.
- If the error was detected within a FRU, the identity of the FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

If the node that owns error record 0 implements a standard format Corrected error counter or counters (ERRFR[FirstRecordOfNode(n)].CEC != `0b000`), then it is **IMPLEMENTATION DEFINED** whether error record 0 can record countable errors, and:

- If error record 0 records countable errors, then ERR0MISC0 implements the standard format Corrected error counter or counters for error record 0.
- If error record 0 does not record countable errors, then it is recommended that the fields in ERR0MISC0 defined for the standard format counter or counters are **RES0**. That is, the fields behave like counters that never count.

Configurations

ERRFR[FirstRecordOfNode(n)] describes the features implemented by the node that owns error record 0. FirstRecordOfNode(n) is the index of the first error record owned by the same node as error record 0. If the node owns a single record then FirstRecordOfNode(n) = n.

For **IMPLEMENTATION DEFINED** fields in ERR0MISC0, writing zero returns the error record to an initial quiescent state.

In particular, if any **IMPLEMENTATION DEFINED** syndrome fields might generate a Fault Handling or Error Recovery Interrupt request, writing zero is sufficient to deactivate the Interrupt request.

Fields that are read-only, nonzero, and ignore writes are compliant with this requirement.

Note

Arm recommends that any **IMPLEMENTATION DEFINED** syndrome field that can generate a Fault Handling, Error Recovery, Critical, or **IMPLEMENTATION DEFINED**, interrupt request is disabled at Cold reset and is enabled by software writing an **IMPLEMENTATION DEFINED** nonzero value to an **IMPLEMENTATION DEFINED** field in ERRCTLR[FirstRecordOfNode(n)].

Attributes

Width

64

Component

RAS

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Register offset

0x20

Access type

RW

Reset value

```
xxxx xxx0 0000 0000 0000 0000 0000 0000 xxxx x000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-45: EXT_ERR0MISC0 bit assignments

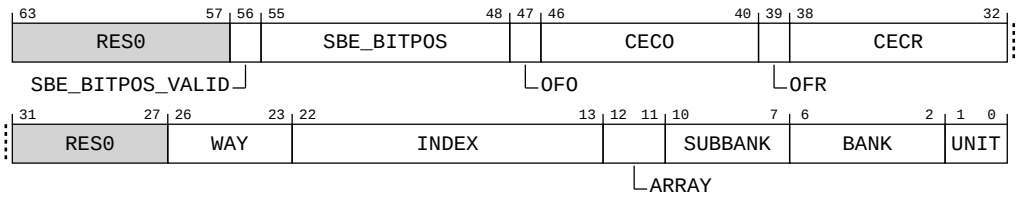


Table B-94: ERR0MISC0 bit descriptions

Bits	Name	Description	Reset
[63:57]	**RES0**	Reserved	**RES0**
[56]	SBE_BITPOS_VALID	Bit postion of a corrected error from a RAM protected by ECC is Valid.	`0b0`
[55:48]	SBE_BITPOS	Bit postion of a corrected error from a RAM protected by ECC. This field is used for Data RAM, Tag RAM and Context RAM.	`0x00`
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERR0MISC0.CECO is incremented and wraps through zero. **`0b0`** Other counter has not overflowed. **`0b1`** Other counter has overflowed. A direct write that modifies this bit might indirectly set ERR0STATUS.OF to an **UNKNOWN** value and a direct write to ERR0STATUS.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Warm reset.	`0b0`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERR0MISC0.CECR. Unaffected by Warm reset.	`0b0000000`
[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERR0MISC0.CECR is incremented and wraps through zero. **`0b0`** Repeat counter has not overflowed. **`0b1`** Repeat counter has overflowed. A direct write that modifies this bit might indirectly set ERR0STATUS.OF to an **UNKNOWN** value and a direct write to ERR0STATUS.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Warm reset.	`0b0`
[38:32]	CECR	Corrected error count, repeat. Incremented for the first countable error, which also records other syndrome for the error, and subsequently for each countable error that matches the recorded other syndrome.	`0b0000000`
[31:27]	**RES0**	Reserved	**RES0**
[26:23]	WAY	Indicates which way detected the error. This field is used for Tag RAM and Data RAM. Upper 2 bits are unused. This field is zero for Context RAM. Unaffected by Warm reset.	`0b0000`
[22:13]	INDEX	Indicates which index detected the error: • For Data RAM and Tag RAM upper 4 bits are unused • For Context RAM all bits are used Unaffected by Warm reset.	`0b0000000000`
[12:11]	ARRAY	Indicates which array detected the error. The possible values are: • 00 Tag RAM. • 01 Data RAM. • 10 Context RAM. Unaffected by Warm reset.	`0b00`
[10:7]	SUBBANK	Indicates which subbank detected the error. This field is used for Data RAM and Context RAM. Upper 2 bits are unused. This field is zero for Tag RAM. Unaffected by Warm reset.	`0b0000`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[6:2]	BANK	Indicates which bank detected the error. This field is used for Data Ram, Tag Ram and Context RAM. Upper 2 bits are unused. Bits [2:0] are encoded in such a way: • bit [2]: Superbank number. There are up to two superbanks • bit [1]: Pipe number. There are up to two pipes for each superbank • bit [0]: Bank number. This bit is zero as there is one single bank for each pipe Unaffected by Warm reset.	`0b00000`
[1:0]	UNIT	Indicates the unit which detected the error. The possible values are: **`0b10`** L1 Cache.	`0b00`

Accessibility

Reads from ERR0MISC0 return an **IMPLEMENTATION DEFINED** value and writes have **IMPLEMENTATION DEFINED** behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ERR0STATUS.MV is 0. See ERR0PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.
- When ERR0STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.

Note

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Component	Offset	Instance	Range
RAS	0x20	ERR0MISC0	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.3.6 ERR0MISC1, Error Record <n> Miscellaneous Register 1

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to locate where the error was detected.
- If the error was detected within a FRU, the identity of the FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

Configurations

ERRFR[FirstRecordOfNode(n)] describes the features implemented by the node that owns error record 0. FirstRecordOfNode(n) is the index of the first error record owned by the same node as error record 0. If the node owns a single record then FirstRecordOfNode(n) = n.

For **IMPLEMENTATION DEFINED** fields in ERR0MISC1, writing zero returns the error record to an initial quiescent state.

In particular, if any **IMPLEMENTATION DEFINED** syndrome fields might generate a Fault Handling or Error Recovery Interrupt request, writing zero is sufficient to deactivate the Interrupt request.

Fields that are read-only, nonzero, and ignore writes are compliant with this requirement.

Note

Arm recommends that any **IMPLEMENTATION DEFINED** syndrome field that can generate a Fault Handling, Error Recovery, Critical, or **IMPLEMENTATION DEFINED**, interrupt request is disabled at Cold reset and is enabled by software writing an **IMPLEMENTATION DEFINED** nonzero value to an **IMPLEMENTATION DEFINED** field in ERRCTLR[FirstRecordOfNode(n)].

Attributes

Width

64

Component

RAS

Register offset

0x28

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-46: EXT_ERR0MISC1 bit assignments

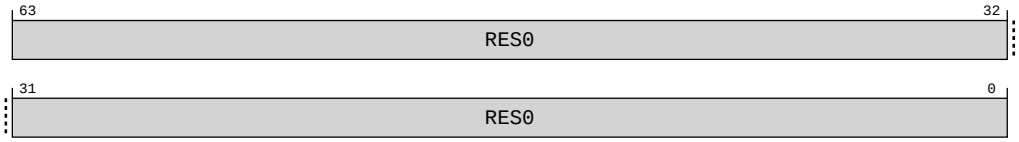


Table B-96: ERR0MISC1 bit descriptions

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

Accessibility

Reads from ERR0MISC1 return an **IMPLEMENTATION DEFINED** value and writes have **IMPLEMENTATION DEFINED** behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ERR0STATUS.MV is 0. See ERR0PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.
- When ERR0STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.

Note

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Component	Offset	Instance	Range
RAS	0x28	ERR0MISC1	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.7 ERR0MISC2, Error Record <n> Miscellaneous Register 2

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to locate where the error was detected.
- If the error was detected within a FRU, the identity of the FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

Configurations

ERRFR[FirstRecordOfNode(n)] describes the features implemented by the node that owns error record 0. FirstRecordOfNode(n) is the index of the first error record owned by the same node as error record 0. If the node owns a single record then FirstRecordOfNode(n) = n.

For **IMPLEMENTATION DEFINED** fields in ERR0MISC2, writing zero returns the error record to an initial quiescent state.

In particular, if any **IMPLEMENTATION DEFINED** syndrome fields might generate a Fault Handling or Error Recovery Interrupt request, writing zero is sufficient to deactivate the Interrupt request.

Fields that are read-only, nonzero, and ignore writes are compliant with this requirement.

Arm recommends that if RAS System Architecture v1.1 is not implemented then ERR0MISC2 does not require zeroing to return the record to a quiescent state.

Note

Arm recommends that any **IMPLEMENTATION DEFINED** syndrome field that can generate a Fault Handling, Error Recovery, Critical, or **IMPLEMENTATION DEFINED**, interrupt request is disabled at Cold reset and is enabled by software writing an **IMPLEMENTATION DEFINED** nonzero value to an **IMPLEMENTATION DEFINED** field in ERRCTLR[FirstRecordOfNode(n)].

Attributes

Width

64

Component

RAS

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Register offset

0x30

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-47: EXT_ERR0MISC2 bit assignments

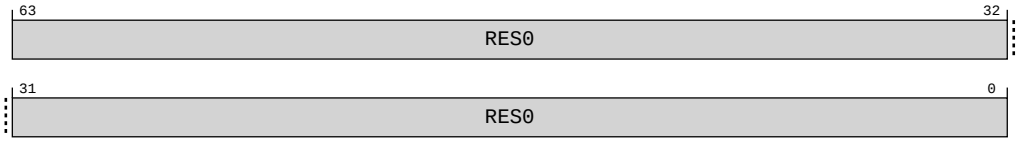


Table B-98: ERR0MISC2 bit descriptions

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

Accessibility

Reads from ERR0MISC2 return an **IMPLEMENTATION DEFINED** value and writes have **IMPLEMENTATION DEFINED** behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ERR0STATUS.MV is 0. See ERR0PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.
- When ERR0STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Note

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Component	Offset	Instance	Range
RAS	0x30	ERR0MISC2	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.8 ERR0MISC3, Error Record <n> Miscellaneous Register 3

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to locate where the error was detected.
- If the error was detected within a FRU, the identity of the FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

If the node that owns error record n supports the RAS Timestamp Extension (ERRFR[FirstRecordOfNode(n)].TS != `0b00`), then ERR0MISC3 contains the timestamp value for error record n when the error was detected. Otherwise the contents of ERR0MISC3 are **IMPLEMENTATION DEFINED**.

Configurations

ERRFR[FirstRecordOfNode(n)] describes the features implemented by the node that owns error record 0. FirstRecordOfNode(n) is the index of the first error record owned by the same node as error record 0. If the node owns a single record then FirstRecordOfNode(n) = n.

For **IMPLEMENTATION DEFINED** fields in ERR0MISC3, writing zero returns the error record to an initial quiescent state.

In particular, if any **IMPLEMENTATION DEFINED** syndrome fields might generate a Fault Handling or Error Recovery Interrupt request, writing zero is sufficient to deactivate the Interrupt request.

Fields that are read-only, nonzero, and ignore writes are compliant with this requirement.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Arm recommends that if RAS System Architecture v1.1 is not implemented then ERR0MISC3 does not require zeroing to return the record to a quiescent state.

Note

Arm recommends that any **IMPLEMENTATION DEFINED** syndrome field that can generate a Fault Handling, Error Recovery, Critical, or **IMPLEMENTATION DEFINED**, interrupt request is disabled at Cold reset and is enabled by software writing an **IMPLEMENTATION DEFINED** nonzero value to an **IMPLEMENTATION DEFINED** field in ERRCTLR[FirstRecordOfNode(n)].

Attributes

Width

64

Component

RAS

Register offset

0x38

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-48: EXT_ERR0MISC3 bit assignments

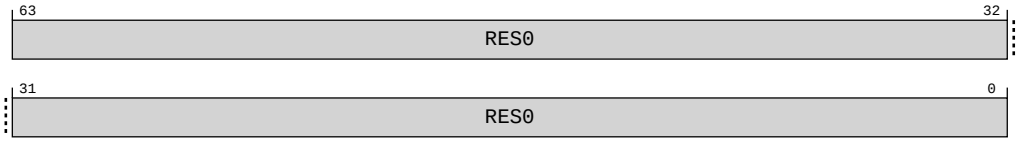


Table B-100: ERR0MISC3 bit descriptions

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

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

Reads from ERR0MISC3 return an **IMPLEMENTATION DEFINED** value and writes have **IMPLEMENTATION DEFINED** behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ERR0STATUS.MV is 0. See ERR0PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.
- When ERR0STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.

Note

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Component	Offset	Instance	Range
RAS	0x38	ERR0MISC3	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.9 ERR0PFGF, Error Record <n> Pseudo-fault Generation Feature Register

Defines which common architecturally-defined fault generation features are implemented.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Component

RAS

Register offset

0x800

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0111 0000 0000 0000 0001 1010 0110 0011
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-49: EXT_ERR0PFGF bit assignments

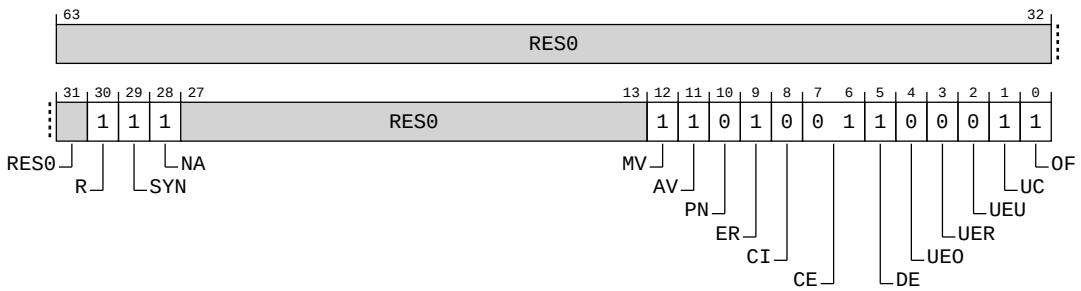


Table B-102: ERR0PFGF bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30]	R	Restartable. Support for Error Generation Counter restart mode. **0b1** Error Generation Counter restart mode is implemented and is controlled by ERR0PFGCTL.R. ERR0PFGCTL.R is a read/write field.	0b1
[29]	SYN	Syndrome. Fault syndrome injection. **0b1** When an injected error is recorded, the node does not update the ERR0STATUS.SERR field. ERR0STATUS.SERR is writable when ERR0STATUS.V is 0.	0b1
[28]	NA	No access required. Defines whether this component fakes detection of the error on an access to the component or spontaneously in the fault injection state. **0b1** The component fakes detection of the error spontaneously in the fault injection state.	0b1
[27:13]	**RES0**	Reserved	**RES0**

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[12]	MV	Miscellaneous syndrome. Defines whether software can control all or part of the syndrome recorded in the ERR0MISC<m> registers when an injected error is recorded. It is **IMPLEMENTATION DEFINED** which ERR0MISC<m> syndrome fields, if any, are updated by the node when an injected error is recorded. Some syndrome fields might always be updated by the node when an error, including an injected error, is recorded. For example, a corrected error counter might always be updated when any countable error, including a injected countable error, is recorded. **0b1** When an injected error is recorded, the node does not updates ERR0MISC<m> and ERR0STATUS.MV is set to ERR0PFGCTL.MV. ERR0PFGCTL.MV is a read/write field.	`0b1`
[11]	AV	Address syndrome. Defines whether software can control the address recorded in ERR0ADDR when an injected error is recorded. **0b1** When an injected error is recorded, the node does not update ERR0ADDR and ERR0STATUS.AV is set to ERR0PFGCTL.AV. ERR0PFGCTL.AV a read/write field.	`0b1`
[10]	PN	Poison flag. Describes how the fault generation feature of the node sets the ERR0STATUS.PN status flag. **0b0** When an injected error is recorded the node sets ERR0STATUS.PN to 0. ERR0PFGCTL.PN is **RES0**.	`0b0`
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the ERR0STATUS.ER status flag. **0b1** When an injected error is recorded, ERR0STATUS.ER is set to ERR0PFGCTL.ER. This behavior replaces the architecture-defined rules for setting the ER field. ERR0PFGCTL.ER is a read/write field.	`0b1`
[8]	CI	Critical Error flag. Describes how the fault generation feature of the node sets the ERR0STATUS.CI status flag. **0b0** When an injected error is recorded, the node sets ERR0STATUS.CI to 0. ERR0PFGCTL.CI is **RES0**.	`0b0`
[7:6]	CE	Corrected Error generation. Describes the types of Corrected error that the fault generation feature of the node can generate. **0b01** The fault generation feature of the node allows generation of a non-specific Corrected error, that is, a Corrected error that is recorded by setting ERR0STATUS.CE to `0b10`. ERR0PFGCTL.CE is a read/write field. The values `0b10` and `0b11` in ERR0PFGCTL.CE are reserved.	`0b01`
[5]	DE	Deferred Error generation. Describes whether the fault generation feature of the node can generate Deferred errors. **0b1** The fault generation feature of the node allows generation of Deferred errors. ERR0PFGCTL.DE is a read/write field.	`0b1`
[4]	UEO	Latent or Restartable Error generation. Describes whether the fault generation feature of the node can generate Latent or Restartable errors. **0b0** The fault generation feature of the node does not generate Latent or Restartable errors. ERR0PFGCTL.UEO is **RES0**.	`0b0`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[3]	UER	Signaled or Recoverable Error generation. Describes whether the fault generation feature of the node can generate Signaled or Recoverable errors. **0b0** The fault generation feature of the node does not generate Signaled or Recoverable errors. ERR0PFGCTL.UER is **RES0**.	0b0
[2]	UEU	Unrecoverable Error generation. Describes whether the fault generation feature of the node can generate Unrecoverable errors. **0b0** The fault generation feature of the node does not generate Unrecoverable errors. ERR0PFGCTL.UEU is **RES0**.	0b0
[1]	UC	Uncontainable Error generation. Describes whether the fault generation feature of the node can generate Uncontainable errors. **0b1** The fault generation feature of the node allows generation of Uncontainable errors. ERR0PFGCTL.UC is a read/write field.	0b1
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the ERR0STATUS.OF status flag. **0b1** When an injected error is recorded, ERR0STATUS.OF is set to ERR0PFGCTL.OF. This behavior replaces the architecture-defined rules for setting the OF field. ERR0PFGCTL_EL1.OF is a read/write field.	0b1

Accessibility

Component	Offset	Instance	Range
RAS	0x800	ERR0PFGF	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.10 ERR0PFGCTL, Error Record <n> Pseudo-fault Generation Control Register

Enables controlled fault generation.

Configurations

ERR0PFGF describes the Common Fault Injection features implemented by the node.

ERR0FR describes the features implemented by the node.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

64

Component

RAS

Register offset

0x808

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-50: EXT_ERR0PFGCTL bit assignments

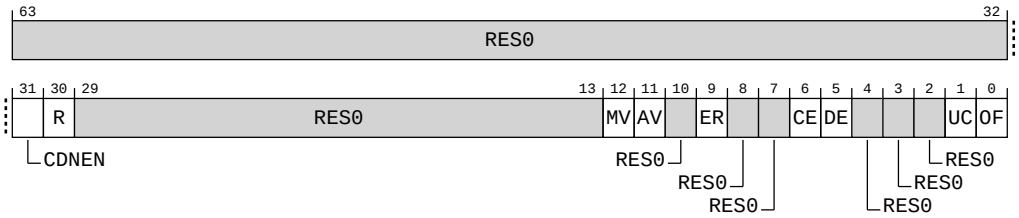


Table B-104: ERR0PFGCTL bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	CDNEN	Countdown Enable. Controls transfers of the value held in ERR0PFGCDN to the Error Generation Counter and enables this counter. **0b0** The Error Generation Counter is disabled. **0b1** The Error Generation Counter is enabled. On a write of 1 to this field, the Error Generation Counter is set to ERR0PFGCDN.CDN.	0b0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[30]	R	Restart. Controls whether the Error Generation Counter restarts or stops counting on reaching zero. **0b0** On reaching zero, the Error Generation Counter will stop counting. **0b1** On reaching zero, the Error Generation Counter is set to ERR0PFGCDN.CDN.	x
[29:13]	**RES0**	Reserved	**RES0**
[12]	MV	Miscellaneous syndrome. The value written to ERR0STATUS.MV when an injected error is recorded. **0b0** ERR0STATUS.MV is set to 0 when an injected error is recorded. **0b1** ERR0STATUS.MV is set to 1 when an injected error is recorded.	x
[11]	AV	Address syndrome. The value written to ERR0STATUS.AV when an injected error is recorded. **0b0** ERR0STATUS.AV is set to 0 when an injected error is recorded. **0b1** ERR0STATUS.AV is set to 1 when an injected error is recorded.	x
[10]	**RES0**	Reserved	**RES0**
[9]	ER	Error Reported flag. The value written to ERR0STATUS.ER when an injected error is recorded. **0b0** ERR0STATUS.ER is set to 0 when an injected error is recorded. **0b1** ERR0STATUS.ER is set to 1 when an injected error is recorded.	x
[8:7]	**RES0**	Reserved	**RES0**
[6]	CE	Corrected Error generation enable. Controls the type of injected Corrected error generated by the fault injection feature of the node. **0b0** An injected Corrected error will not be generated by the fault injection feature of the node. **0b1** An injected non-specific Corrected error is generated in the fault injection state. ERR<n>STATUS.CE is set to `0b10` when the injected error is recorded. The set of permitted values for this field is defined by ERR<n>PFGF.CE. The node enters the fault injection state when the Error Generation Counter decrements to zero.	x
[5]	DE	Deferred Error generation enable. Controls whether an injected Deferred error is generated by the fault injection feature of the node. **0b0** An injected Deferred error will not be generated by the fault generation feature of the node. **0b1** An injected Deferred error is generated in the fault injection state. The node enters the fault injection state when the Error Generation Counter decrements to zero	x
[4:2]	**RES0**	Reserved	**RES0**

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[1]	UC	Uncontainable Error generation enable. Controls whether an injected Uncontainable error is generated by the fault injection feature of the node. **0b0** An injected Uncontainable error will not be generated by the fault generation feature of the node. **0b1** An injected Uncontainable error is generated in the fault injection state. The node enters the fault injection state when the Error Generation Counter decrements to zero	x
[0]	OF	Overflow flag. The value written to ERR0STATUS.OF when an injected error is recorded. **0b0** ERR0STATUS.OF is set to 0 when an injected error is recorded. **0b1** ERR0STATUS.OF is set to 1 when an injected error is recorded.	x

Accessibility

Component	Offset	Instance	Range
RAS	0x808	ERR0PFGCTL	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.11 ERR0PFGCDN, Error Record <n> Pseudo-fault Generation Countdown Register

Generates one of the errors enabled in the corresponding ERR0PFGCTL register.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

RAS

Register offset

0x810

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-51: EXT_ERR0PFGCDN bit assignments

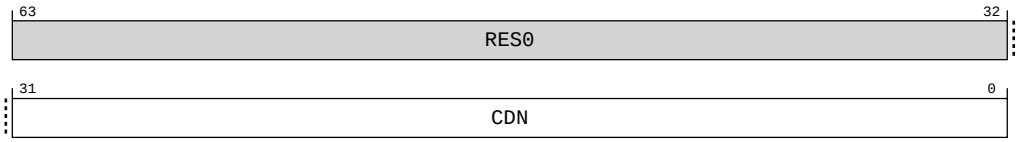


Table B-106: ERR0PFGCDN bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	CDN	Countdown value. This field is copied to Error Generation Counter when either Software writes ERR0PFGCTL.CDNEN with 1 or The Error Generation Counter decrements to zero and ERR0PFGCTL.R == 1. **Note:** The current Error Generation Counter value is not visible to software.	32{x}

Accessibility

Component	Offset	Instance	Range
RAS	0x810	ERR0PFGCDN	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RW

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.3.12 ERRGSR, Error Group Status Register

Shows the status for the records in the group.

Configurations

ERRGSR is implemented only as part of a memory-mapped group of error records.

This manual describes a group of error records accessed via a standard 4KB memory-mapped peripheral. For a 4KB peripheral, up to 24 error records can be accessed if the Common Fault Injection Model is implemented, and up to 56 otherwise.

Attributes

Width

64

Component

RAS

Register offset

0xE00

Access type

RO

Reset value

```
0000 0000 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-52: EXT_ERRGSR bit assignments

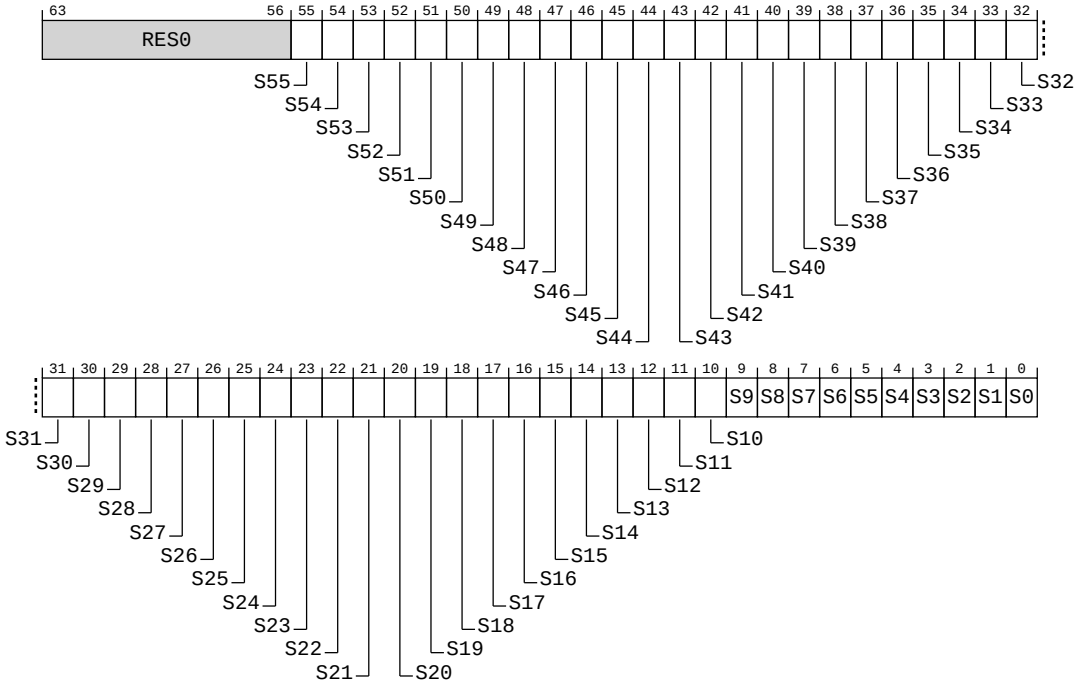


Table B-108: ERRGSR bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[55:0]	S<m>, bit[m], where m = 55 to 0	The status for error record <m>. A read-only copy of ERR<m>STATUS.V. **0b0** No error. **0b1** One or more errors. If the Common Fault Injection Model is implemented then up-to 24 records can be implemented meaning bits [55:24] are **RES0**.	`56{x}`

Accessibility

Component	Offset	Instance	Range
RAS	0xE00	ERRGSR	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Otherwise

ERROR

B.3.13 ERRIIDR, Implementation Identification Register

Defines the implementer of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

RAS

Register offset

0xE10

Access type

RO

Reset value

```
1101 1000 1101 0001 0010 0100 0011 1011
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-53: EXT_ERRIIDR bit assignments

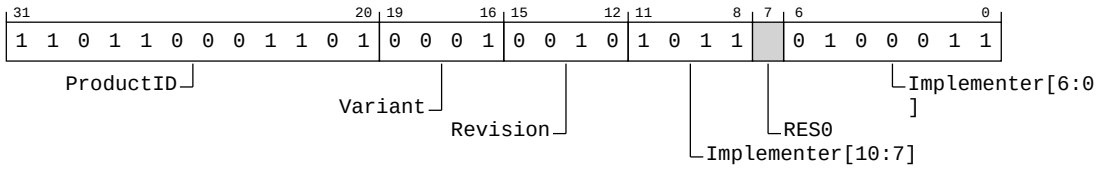


Table B-110: ERRIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	Part number, bits [11:0]. The part number is selected by the designer of the component. **0xD8D** C1-SME2	0xD8D

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[19:16]	Variant	Component major revision. This field distinguishes product variants or major revisions of the product. **`0b0001`** r1p2	`0b0001`
[15:12]	Revision	Component minor revision. This field distinguishes minor revisions of the product. **`0b0010`** r1p2	`0b0010`
[11:8, 6:0]	Implementer	Contains the JEP106 code of the company that implemented the RAS component. For an Arm implementation, this field has the value `0x43B`. **`0b01000111011`** Arm Limited	`0b01000111011`
[7]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Range
RAS	0xE10	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.14 ERRDEVAFF, Device Affinity Register

Device Affinity for the RAS.

Configurations

ERRDEVAFF is implemented only as part of a memory-mapped group of error records.

Attributes

Width

64

Component

RAS

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Register offset

0xFA8

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 xxxx xxxx 0x00 000x xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-54: EXT_ERRDEVAFF bit assignments

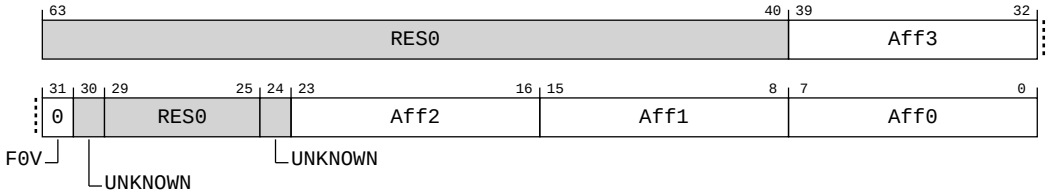


Table B-112: ERRDEVAFF bit descriptions

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	RES0
[39:32]	Aff3	PE affinity level 3. **0x00** Affinity 3 not supported.	8{x}
[31]	F0V	Indicates that the ERRDEVAFF.Aff0 field is valid. **0b0** ERRDEVAFF.Aff0 is not valid, and the PE affinity is above level 0 or a subset of level 0.	0b0
[30]	UNKNOWN	Reserved	UNKNOWN
[29:25]	RES0	Reserved	RES0
[24]	UNKNOWN	Reserved	UNKNOWN
[23:16]	Aff2	PE affinity level 2. **0x00** Affinity 2 not supported.	8{x}
[15:8]	Aff1	PE affinity level 1. **0x00** Affinity 1 not supported.	8{x}

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[7:0]	Aff0	PE affinity level 0. **0x00** Affinity 0 not supported.	8 { x }

Accessibility

Component	Offset	Instance	Range
RAS	0xFA8	ERRDEVAFF	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.15 ERRDEVARCH, Device Architecture Register

Provides discovery information for the component.

Configurations

ERRDEVARCH is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

Register offset

0xFBC

Access type

RO

Reset value

```
0100 0111 0111 0001 0000 1010 0000 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-55: EXT_ERRDEVARCH bit assignments

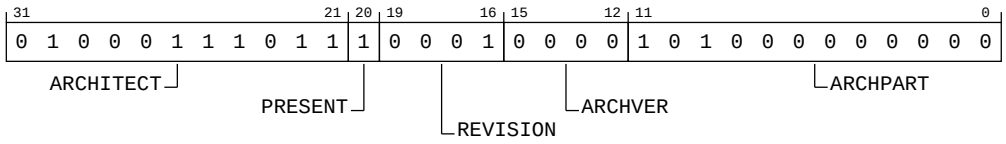


Table B-114: ERRDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architect of the component. For RAS, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, `0b0100`. Bits [27:21] are the JEP106 identification code, `0b0111011`. **`0b01000111011`**	`0b01000111011`
[20]	PRESENT	DEVARCH present. Indicates that the ERRDEVARCH register is present. **`0b1`**	`0b1`
[19:16]	REVISION	Revision. Defines the architecture revision of the component. **`0b0001`** RAS System Architecture, error record group v1.1. As `0b0000` and also: • Simplifies ERR<n>STATUS. • Adds support for additional ERR<n>MISC<m> registers. • Adds support for the optional RAS Timestamp Extension. • Adds support for the optional Common Fault Injection Model Extension. All other values are reserved.	`0b0001`
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **`0b0000`** RAS System Architecture, error record group v1. All other values are reserved. ERRDEVARCH.ARCHVER and ERRDEVARCH.ARCHPART are also defined as a single field, ERRDEVARCH.ARCHID, so that ERRDEVARCH.ARCHVER is ERRDEVARCH.ARCHID[15:12].	`0b0000`
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **`0xA00`** RAS System Architecture, error record group. ERRDEVARCH.ARCHVER and ERRDEVARCH.ARCHPART are also defined as a single field, ERRDEVARCH.ARCHID, so that ERRDEVARCH.ARCHPART is ERRDEVARCH.ARCHID[11:0].	`0xA00`

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

Component	Offset	Instance	Range
RAS	0xFBC	ERRDEVARCH	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.16 ERRDEVID, Device Configuration Register

Provides discovery information for the component.

Configurations

ERRDEVID is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

Register offset

0xFC8

Access type

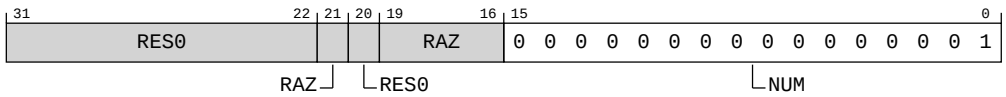
RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0001
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-56: EXT_ERRDEVID bit assignments



Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table B-116: ERRDEVID bit descriptions

Bits	Name	Description	Reset
[31:22]	**RES0**	Reserved	**RES0**
[21]	**RAZ**	Reserved	**RAZ**
[20]	**RES0**	Reserved	**RES0**
[19:16]	**RAZ**	Reserved	**RAZ**
[15:0]	NUM	Highest numbered index of the error records in this group, plus one. Each implemented record is owned by a node. A node might own multiple records. **`0x0001`** One record present.	`0x0001`

Accessibility

Component	Offset	Instance	Range
RAS	0xFC8	ERRDEVID	None

This interface is accessible as follows:

RO

B.3.17 ERRPIDR4, Peripheral Identification Register 4

Provides discovery information about the component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

ERRPIDR4 is implemented only as part of a memory-mapped group of error records.

This register is present only when ImpDefBool("IMPLEMENTED_ERRPIDR4. Otherwise, direct accesses to ERRPIDR4 are UNDEFINED.

Attributes

Width

32

Component

RAS

Register offset

0xFD0

Access type

RO

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-57: EXT_ERRPIDR4 bit assignments

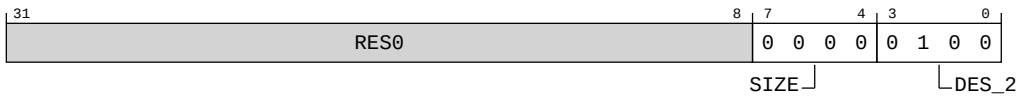


Table B-118: ERRPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	Size of the component. The distance from the start of the address space used by this component to the end of the component identification registers. A value of `0b0000` means one of the following is true: • The component uses a single 4KB block. • The component uses an **IMPLEMENTATION DEFINED** number of 4KB blocks. Any other value means the component occupies $2^{\text{ERRPIDR4.SIZE}}$ 4KB blocks. **`0b0000`** The component uses a single 4KB block	`0b0000`
[3:0]	DES_2	Designer, JEP106 continuation code. This is the JEDEC-assigned JEP106 bank identifier for the designer of the component, minus 1. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. **`0b0100`** Arm Limited	`0b0100`

Accessibility

Component	Offset	Instance	Range
RAS	0xFD0	ERRPIDR4	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Otherwise

ERROR

B.3.18 ERRPIDR0, Peripheral Identification Register 0

Provides discovery information about the component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

ERRPIDR0 is implemented only as part of a memory-mapped group of error records.

This register is present only when ImpDefBool("IMPLEMENTED_ERRPIDR0. Otherwise, direct accesses to ERRPIDR0 are UNDEFINED.

Attributes

Width

32

Component

RAS

Register offset

0xFE0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1000 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-58: EXT_ERRPIDR0 bit assignments

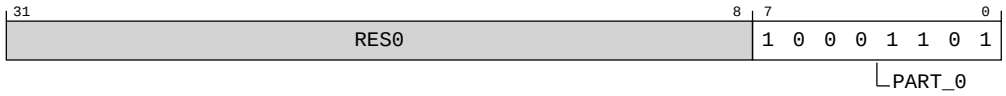


Table B-120: ERRPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[7:0]	PART_0	Part number, bits [7:0]. The part number is selected by the designer of the component. The designer chooses whether to use a 12-bit or a 16-bit part number: • If a 12-bit part number is used, then it is stored in ERRPIDR1.PART_1 and ERRPIDR0.PART_0. There are 8 bits, ERRPIDR2.REVISION and ERRPIDR3.REVAND, available to define the revision of the component. • If a 16-bit part number is used, then it is stored in ERRPIDR2.PART_2, ERRPIDR1.PART_1 and ERRPIDR0.PART_0. There are 4 bits, ERRPIDR3.REVISION, available to define the revision of the component. **0x8D** C1-SME2	0x8D

Accessibility

Component	Offset	Instance	Range
RAS	0xFE0	ERRPIDR0	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.19 ERRPIDR1, Peripheral Identification Register 1

Provides discovery information about the component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

ERRPIDR1 is implemented only as part of a memory-mapped group of error records.

This register is present only when ImpDefBool("IMPLEMENTED_ERRPIDR1. Otherwise, direct accesses to ERRPIDR1 are UNDEFINED.

Attributes

Width

32

Component

RAS

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Register offset

0xFE4

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1011 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-59: EXT_ERRPIDR1 bit assignments

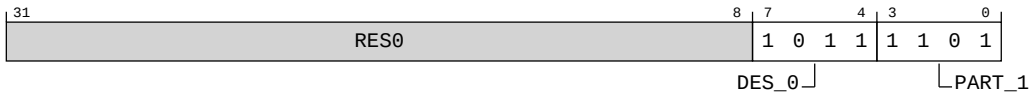


Table B-122: ERRPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, JEP106 identification code, bits [3:0]. ERRPIDR1.DES_0 and ERRPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. **0b1011** Arm Limited	0b1011
[3:0]	PART_1	Part number, bits [11:8]. The part number is selected by the designer of the component. The designer chooses whether to use a 12-bit or a 16-bit part number: • If a 12-bit part number is used, then it is stored in ERRPIDR1.PART_1 and ERRPIDR0.PART_0. There are 8 bits, ERRPIDR2.REVISION and ERRPIDR3.REVAND, available to define the revision of the component. • If a 16-bit part number is used, then it is stored in ERRPIDR2.PART_2, ERRPIDR1.PART_1 and ERRPIDR0.PART_0. There are 4 bits, ERRPIDR3.REVISION, available to define the revision of the component. **0b1101** C1-SME2	0b1101

Accessibility

Component	Offset	Instance	Range
RAS	0xFE4	ERRPIDR1	None

This interface is accessible as follows:

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.20 ERRPIDR2, Peripheral Identification Register 2

Provides discovery information about the component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

ERRPIDR2 is implemented only as part of a memory-mapped group of error records.

This register is present only when ImpDefBool("IMPLEMENTED_ERRPIDR2. Otherwise, direct accesses to ERRPIDR2 are UNDEFINED.

Attributes

Width

32

Component

RAS

Register offset

0xFE8

Access type

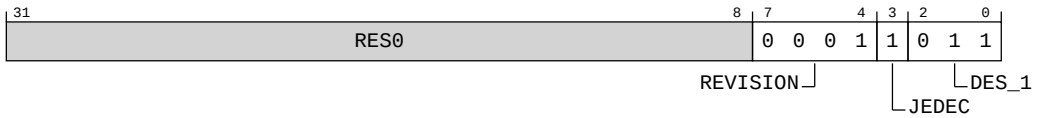
RO

Reset value

```
0000 0000 0000 0000 0000 0000 0001 1011
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-60: EXT_ERRPIDR2 bit assignments



Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table B-124: ERRPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Component major revision. ERRPIDR2.REVISION and ERRPIDR3.REVAND together form the revision number of the component, with ERRPIDR2.REVISION being the most significant part and ERRPIDR3.REVAND the least significant part. When a component is changed, ERRPIDR2.REVISION or ERRPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. ERRPIDR3.REVAND should be set to `0b0000` when ERRPIDR2.REVISION is increased. **`0b0001`**	`0b0001`
[3]	JEDEC	JEDEC-assigned JEP106 implementer code is used. **`0b1`**	`0b1`
[2:0]	DES_1	Designer, JEP106 identification code, bits [6:4]. ERRPIDR1.DES_0 and ERRPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. **`0b011`** Arm Limited	`0b011`

Accessibility

Component	Offset	Instance	Range
RAS	0xFE8	ERRPIDR2	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.21 ERRPIDR3, Peripheral Identification Register 3

Provides discovery information about the component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

ERRPIDR3 is implemented only as part of a memory-mapped group of error records.

This register is present only when ImpDefBool("IMPLEMENTED_ERRPIDR3. Otherwise, direct accesses to ERRPIDR3 are UNDEFINED.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

32

Component

RAS

Register offset

0xFEC

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0010 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-61: EXT_ERRPIDR3 bit assignments

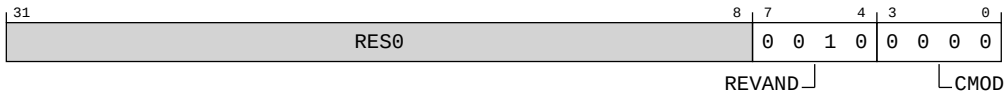


Table B-126: ERRPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Component minor revision. ERRPIDR2.REVISION and ERRPIDR3.REVAND together form the revision number of the component, with ERRPIDR2.REVISION being the most significant part and ERRPIDR3.REVAND the least significant part. When a component is changed, ERRPIDR2.REVISION or ERRPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. ERRPIDR3.REVAND should be set to `0b0000` when ERRPIDR2.REVISION is increased. **`0b0010`** r1p2	`0b0010`
[3:0]	CMOD	Customer Modified. Indicates the component has been modified. A value of `0b0000` means the component is not modified from the original design. Any other value means the component has been modified in an **IMPLEMENTATION DEFINED** way. **`0b0000`**	`0b0000`

Accessibility

Component	Offset	Instance	Range
RAS	0xFEC	ERRPIDR3	None

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.22 ERRCIDR0, Component Identification Register 0

Provides discovery information about the component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

ERRCIDR0 is implemented only as part of a memory-mapped group of error records.

This register is present only when ImpDefBool("IMPLEMENTED_ERRCIDR0. Otherwise, direct accesses to ERRCIDR0 are UNDEFINED.

Attributes

Width

32

Component

RAS

Register offset

0xFF0

Access type

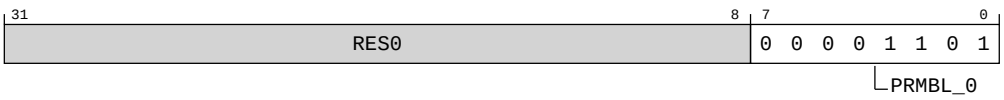
RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-62: EXT_ERRCIDR0 bit assignments



Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table B-128: ERRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Component identification preamble, segment 0. **0x0D**	0x0D

Accessibility

Component	Offset	Instance	Range
RAS	0xFF0	ERRCIDR0	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.23 ERRCIDR1, Component Identification Register 1

Provides discovery information about the component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

ERRCIDR1 is implemented only as part of a memory-mapped group of error records.

This register is present only when ImpDefBool("IMPLEMENTED_ERRCIDR1. Otherwise, direct accesses to ERRCIDR1 are UNDEFINED.

Attributes

Width

32

Component

RAS

Register offset

0xFF4

Access type

RO

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Reset value

```
0000 0000 0000 0000 0000 0000 1111 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-63: EXT_ERRCIDR1 bit assignments

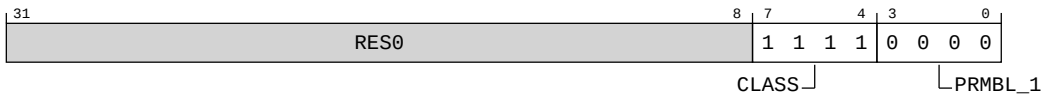


Table B-130: ERRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1111** Generic peripheral with **IMPLEMENTATION DEFINED** register layout. Other values are defined by the CoreSight Architecture. This field reads as `0xF`.	`0b1111`
[3:0]	PRMBL_1	Component identification preamble, segment 1. **0b0000**	`0b0000`

Accessibility

Component	Offset	Instance	Range
RAS	0xFF4	ERRCIDR1	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.24 ERRCIDR2, Component Identification Register 2

Provides discovery information about the component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Configurations

ERRCIDR2 is implemented only as part of a memory-mapped group of error records.

This register is present only when ImpDefBool("IMPLEMENTED_ERRCIDR2. Otherwise, direct accesses to ERRCIDR2 are UNDEFINED.

Attributes

Width

32

Component

RAS

Register offset

0xFF8

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-64: EXT_ERRCIDR2 bit assignments

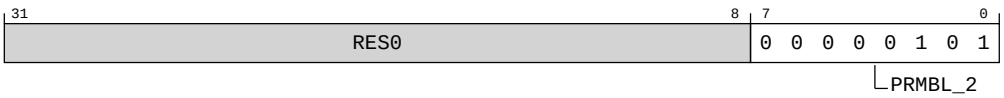


Table B-132: ERRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Component identification preamble, segment 2. **0x05**	0x05

Accessibility

Component	Offset	Instance	Range
RAS	0xFF8	ERRCIDR2	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.3.25 ERRCIDR3, Component Identification Register 3

Provides discovery information about the component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

ERRCIDR3 is implemented only as part of a memory-mapped group of error records.

This register is present only when ImpDefBool("IMPLEMENTED_ERRCIDR3. Otherwise, direct accesses to ERRCIDR3 are UNDEFINED.

Attributes

Width

32

Component

RAS

Register offset

0xFFC

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1011 0001
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-65: EXT_ERRCIDR3 bit assignments

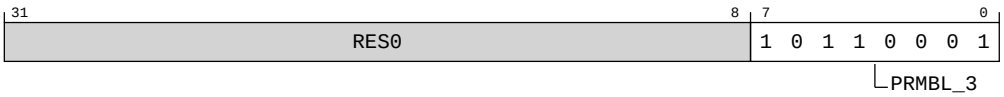


Table B-134: ERRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[7:0]	PRMBL_3	Component identification preamble, segment 3. **0xB1**	0xB1

Accessibility

Component	Offset	Instance	Range
RAS	0xFFC	ERRCIDR3	None

This interface is accessible as follows:

When IsCorePowered() and IsAccessSecure()

RO

When IsCorePowered() and !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.4 External ROM table registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped ROM table registers in the core.

For more information on registers listed in the table, click on the link associated with the register name.

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the *Arm® Architecture Reference Manual for A-profile architecture*.

Table B-136: ROM table registers summary

Offset	Name	Reset	Width	Description
0x0	ROMENTRY0	See individual bit resets.	32-bit	Class 0x9 ROM Table Entries
0xFB8	AUTHSTATUS	See individual bit resets.	32-bit	Authentication Status Register
0xFBC	DEVARCH	0x47700AF7	32-bit	Device Architecture Register
0xFC8	DEVID	See individual bit resets.	32-bit	Device Configuration Register
0xFD0	PIDR4	0x00000004	32-bit	Peripheral Identification Register 4
0xFE0	PIDR0	0x0000008D	32-bit	Peripheral Identification Register 0
0xFE4	PIDR1	0x000000BD	32-bit	Peripheral Identification Register 1
0xFE8	PIDR2	0x0000001B	32-bit	Peripheral Identification Register 2
0xFEC	PIDR3	0x00000020	32-bit	Peripheral Identification Register 3
0xFF0	CIDR0	0x0000000D	32-bit	Component Identification Register 0
0xFF4	CIDR1	0x00000090	32-bit	Component Identification Register 1
0xFF8	CIDR2	0x00000005	32-bit	Component Identification Register 2

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Offset	Name	Reset	Width	Description
0xFFC	CIDR3	`0x000000B1`	32-bit	Component Identification Register 3

B.4.1 ROMENTRY0, Class 0x9 ROM Table Entries

A Class `0x9` ROM Table contains up to 512 ROM Table entries. Each entry that is present, ROMENTRY0, provides the address offset of the address space of one CoreSight component, component 0, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class `0x9` ROM Table:

- The first entry, entry 0, has offset `0x000`.
- ROMENTRY0 has the offset `0x000` + 0×4, where 0 ≤ 0 ≤ 511.
- If the number of components, 0, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:
  - ROM Table entries representing components have offsets from `0x000` to (0-1)×4.
  - The ROMENTRY0 at offset 0×4, which has a PRESENT field with the value `0b00`, indicates the end of the ROM Table.
- If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from `0x000` to `0x7FC`. If a ROM Table entry is present at offset `0x7FC`, its PRESENT field must have a value of either `0b00` or `0b11`, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value `0b11`.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

If DEVID.FORMAT has the value `0x0`, ROMENTRY0 are 512 32-bit registers.

If DEVID.FORMAT has the value `0x1`, ROMENTRY0 are 256 64-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0x0

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx 0000 0000 00xx
```

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-66: EXT_ROMENTRY0 bit assignments

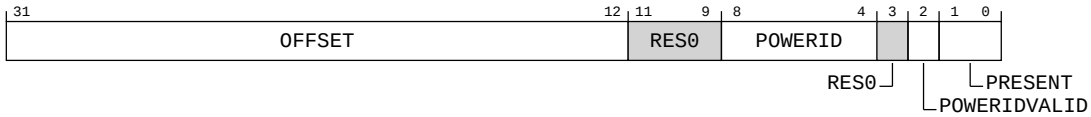


Table B-137: ROMENTRY0 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **0x00000** ROM Entry is not present. **0x00040** Core ELA	`20{x}`
[11:9]	**RES0**	Reserved	**RES0**
[8:4]	POWERID	The power domain ID of the component. This field supports up to 32 power domains using values `0x00` to `0x1F`.	`0b00000`
[3]	**RES0**	Reserved	**RES0**
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0b0** A power domain ID is not provided.	`0b0`
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **0b00** The ROM entry is not present, and this ROMENTRY0 is the final entry in the ROM Table. If PRESENT has this value, all other fields in this ROMENTRY0 must be zero. **0b11** The ROM Entry is present.	The reset values can be the following: `0b00`, `0b11`, respective to the value.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

A ROM Table does not have to use power domain IDs. If none of the ROM Table entries provides a power domain ID, all the components that pointed to by the ROM Table are in the same power domain as the ROM Table.

Component	Offset	Range
ROM table	0x0	None

This interface is accessible as follows:

RO

B.4.2 AUTHSTATUS, Authentication Status Register

AUTHSTATUS indicates whether certain functions are enabled.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFB8

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 xxxx 11xx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bit descriptions

Figure B-67: EXT_AUTHSTATUS bit assignments

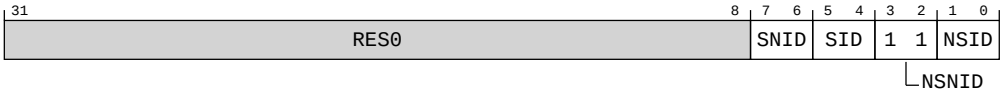


Table B-139: AUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:6]	SNID	Secure Non-invasive Debug. **0b10** Supported and disabled. (SPIDEN & DBGEN) == FALSE. **0b11** Supported and enabled. (SPIDEN & DBGEN) == TRUE.	`xx`
[5:4]	SID	Secure Invasive Debug. **0b10** Supported and disabled. (SPIDEN & DBGEN) == FALSE. **0b11** Supported and enabled. (SPIDEN & DBGEN) == TRUE.	`xx`
[3:2]	NSNID	Non-secure Non-Invasive Debug. **0b11** Debug level supported and enabled	`0b11`
[1:0]	NSID	Non-secure Invasive debug. **0b10** Supported and disabled. DBGEN == FALSE. **0b11** Supported and enabled. DBGEN == TRUE.	`xx`

Accessibility

Component	Offset	Instance	Range
ROM table	0xFB8	AUTHSTATUS	None

This interface is accessible as follows:

RO

B.4.3 DEVARCH, Device Architecture Register

Identifies the architect and architecture of a CoreSight component.

Configurations

This register is available in all configurations.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

32

Component

ROM table

Register offset

0xFBC

Access type

RO

Reset value

```
0100 0111 0111 0000 0000 1010 1111 0111
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-68: EXT_DEVARCH bit assignments

```
31                                  21 20 19        16 15                                  0
0 1 0 0 0 1 1 1 0 1 1 | 1 | 0 0 0 0 | 0 0 0 0 1 0 1 0 1 1 1 1 0 1 1 1
    ARCHITECT                         |     REVISION                ARCHID
                                      PRESENT
```

Table B-141: DEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. **0b01000111011** JEP106 continuation code `0x4`, ID code `0x3B`. Arm Limited.	`0b01000111011`
[20]	PRESENT	Present. **0b1** DEVARCH information present.	`0b1`
[19:16]	REVISION	Revision. **0b0000** Revision 0.	`0b0000`
[15:0]	ARCHID	Architecture ID. **0x0AF7** ROM Table v0. The debug tool must inspect DEVTYPE and DEVID to determine further information about the ROM Table.	`0x0AF7`

Accessibility

Component	Offset	Range
ROM table	0xFBC	None

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

This interface is accessible as follows:

RO

B.4.4 DEVID, Device Configuration Register

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFC8

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-69: EXT_DEVID bit assignments

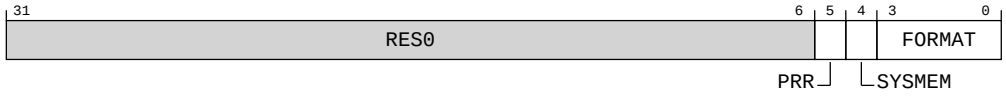


Table B-143: DEVID bit descriptions

Bits	Name	Description	Reset
[31:6]	RES0	Reserved	RES0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[5]	PRR	Power Request functionality not included. **0b0** Power Request functionality not included. If any ROM Table entries contain power domain IDs, a GPR must be present, and pointed to by the ROM Table. The GPR provides functionality to control the power domains. PRIDR0 is not implemented.	0b0
[4]	SYSMEM	System memory present. Indicates whether system memory is present on the bus that connects to the ROM Table. **0b0** System memory is not present on the bus. This value indicates that the bus is a dedicated debug bus. The ROM Table indicates all the valid addresses in the memory system that the ADI is connected to, and the result of accessing any other address is **UNPREDICTABLE**.	0b0
[3:0]	FORMAT	ROM format. **0b0000** 32-bit format 0.	xxxx

Accessibility

Component	Offset	Range
ROM table	0xFC8	None

This interface is accessible as follows:

RO

B.4.5 PIDR4, Peripheral Identification Register 4

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFD0

Access type

RO

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-70: EXT_PIDR4 bit assignments

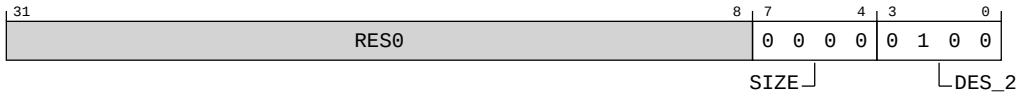


Table B-145: PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	Size of the component. Log_2 of the number of 4KB pages from the start of the component to the end of the component ID registers. **0b0000** A ROM Table occupies a single 4KB block of memory.	0b0000
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **0b0100** Arm Limited	0b0100

Accessibility

Component	Offset	Range
ROM table	0xFD0	None

This interface is accessible as follows:

RO

B.4.6 PIDR0, Peripheral Identification Register 0

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Register offset

0xFE0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1000 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-71: EXT_PIDR0 bit assignments

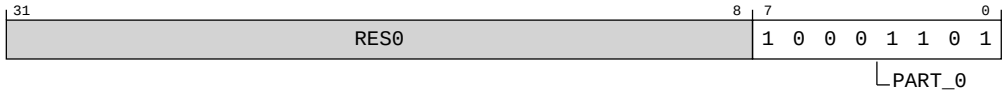


Table B-147: PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0x8D** C1-SME2	0x8D

Accessibility

Component	Offset	Range
ROM table	0xFE0	None

This interface is accessible as follows:

RO

B.4.7 PIDR1, Peripheral Identification Register 1

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Component

ROM table

Register offset

0xFE4

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1011 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-72: EXT_PIDR1 bit assignments

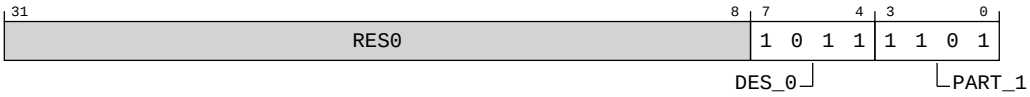


Table B-149: PIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is `0b1011`. **`0b1011`** Arm Limited	`0b1011`
[3:0]	PART_1	Part number, most significant nibble. **`0b1101`** C1-SME2	`0b1101`

Accessibility

Component	Offset	Range
ROM table	0xFE4	None

This interface is accessible as follows:

RO

B.4.8 PIDR2, Peripheral Identification Register 2

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

32

Component

ROM table

Register offset

0xFE8

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0001 1011
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-73: EXT_PIDR2 bit assignments

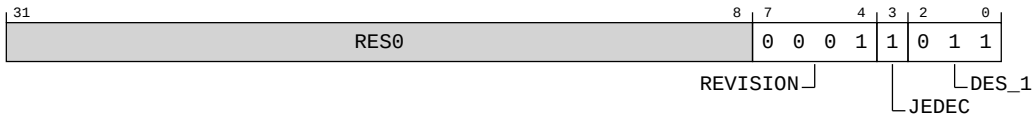


Table B-151: PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0001** r1p2	0b0001
[3]	JEDEC	Indicates a JEP106 identity code is used. **0b1** **RAO**.	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. **0b011** Arm Limited	0b011

Accessibility

Component	Offset	Range
ROM table	0xFE8	None

This interface is accessible as follows:

RO

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

B.4.9 PIDR3, Peripheral Identification Register 3

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFEC

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0010 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-74: EXT_PIDR3 bit assignments

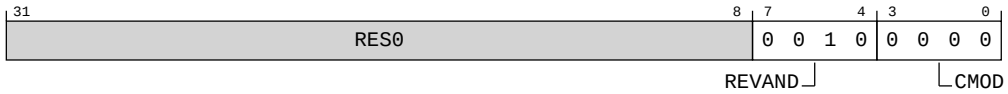


Table B-153: PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Part minor revision. Parts using PIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0010** r1p2	0b0010
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000**	0b0000

Accessibility

Component	Offset	Range
ROM table	0xFEC	None

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

This interface is accessible as follows:

RO

B.4.10 CIDR0, Component Identification Register 0

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFF0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-75: EXT_CIDR0 bit assignments

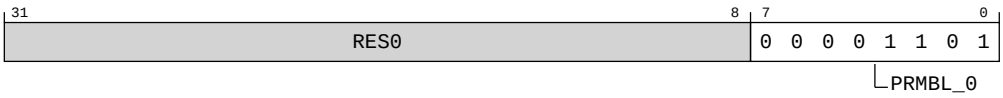


Table B-155: CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	CoreSight component identification preamble. **0x0D** CoreSight component identification preamble.	0x0D

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

Component	Offset	Instance	Range
ROM table	0xFF0	CIDR0	None

This interface is accessible as follows:

RO

B.4.11 CIDR1, Component Identification Register 1

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFF4

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1001 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-76: EXT_CIDR1 bit assignments

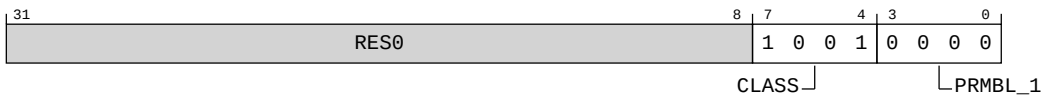


Table B-157: CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	CoreSight component class. **0b1001** CoreSight component.	0b1001

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Bits	Name	Description	Reset
[3:0]	PRMBL_1	CoreSight component identification preamble. **0b0000** CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Range
ROM table	0xFF4	None

This interface is accessible as follows:

RO

B.4.12 CIDR2, Component Identification Register 2

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFF8

Access type

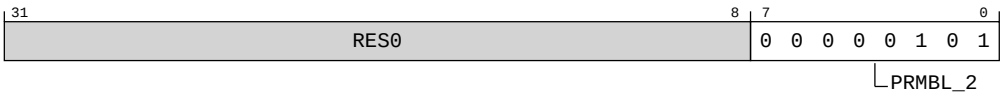
RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-77: EXT_CIDR2 bit assignments



Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table B-159: CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	CoreSight component identification preamble. **0x05** CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Range
ROM table	0xFF8	None

This interface is accessible as follows:

RO

B.4.13 CIDR3, Component Identification Register 3

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFFC

Access type

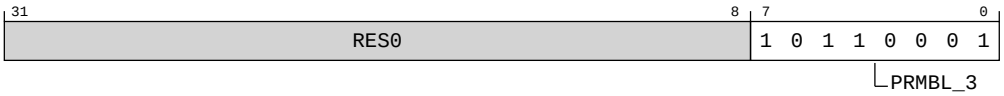
RO

Reset value

```
0000 0000 0000 0000 0000 0000 1011 0001
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-78: EXT_CIDR3 bit assignments



Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table B-161: CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	CoreSight component identification preamble. **`0xB1`** CoreSight component identification preamble.	`0xB1`

Accessibility

Component	Offset	Range
ROM table	0xFFC	None

This interface is accessible as follows:

RO

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Proprietary Notice

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

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

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. 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, any patents, copyrights, trade secrets, trademarks, or other rights.

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.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted 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 this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

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. Please follow Arm’s trademark usage guidelines at https://www.arm.com/company/policies/trademarks. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in the Arm documents.

Product status

All products and Services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

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

Product revision status

This product is r1p2, which indicates the revision status of the product described in this manual, where:

r ***(value)*** Identifies the major revision of the product, for example, r1.

p ***(value)*** Identifies the minor revision or modification status of the product, for example, p2.

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
0102-06	10 September 2025	Non-Confidential	Second early access release for r1p2
0102-05	30 April 2025	Confidential	First early access release for r1p2
0101-04	30 August 2024	Confidential	First early access release for r1p1
0100-03	31 July 2024	Confidential	First early access release for r1p0
0000-02	23 February 2024	Confidential	First limited access release for r0p0

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Issue	Date	Confidentiality	Change
0000-01	24 November 2023	Confidential	First beta release for r0p0

The Change history tables describe the technical changes between released issues of this document in reverse order. Issue numbers match the revision history in Document release information on page 316.

Table 2: Issue 0000-01

Change	Location
First beta release for r0p0	-

Table 3: Differences between issue 0000-01 and issue 0000-02

Change	Location
First limited access release for r0p0	-
Editorial changes	Throughout document
Cortex removed from CME naming	Throughout document
Added missing features	1.4 Supported standards and specifications on page 15
Added a pseudo fault generation counter consideration	4.2.2 Low-power mode behavior considerations on page 33
Added common performance monitoring unit events	12.1 Common performance monitoring unit events on page 65
Added implementation-defined performance monitoring unit events	12.2 Implementation-defined performance monitoring unit events on page 72
Updated Activity monitors events	13.3 Activity monitors events on page 89

Table 4: Differences between issue 0000-02 and issue 0100-03

Change	Location
First early access release for r1p0	-
Editorial changes	Throughout document
Modified low-power descriptions	4.2 Architectural clock gating modes on page 32
Updated power mode information	4.4.1 On mode on page 36
Updated security state for quility bus base addresses for system component registers	9.1 Base addresses for system components on page 54
Modified clock gate and power control information	4.3 Power control on page 33
Updated CoreSight component identification table values	11.4 CoreSight component identification on page 63
Updated common performance monitoring unit events	12.1 Common performance monitoring unit events on page 65
Updated implementation-defined performance monitoring unit events	12.2 Implementation-defined performance monitoring unit events on page 72
Modified bit field names in IMP_CMEPWR_EL1, CME Power Register	A.1.6 IMP_CMEPWR_EL1, SME2 Power Register on page 106

Table 5: Differences between issue 0100-03 and issue 0101-04

Change	Location
First early access release for r1p1	-
Updated r1p0 to r1p1	Throughout document

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table 6: Differences between issue 0101-04 and issue 0102-05

Change	Location
First early access release for r1p2	-
Updated r1p1 to r1p2	Throughout document
Updated PMU events	12. Performance Monitors Extension support on page 65

Table 7: Differences between issue 0102-05 and issue 0102-06

Change	Location
Second early access release for r1p2	-
Updated product name to C1-SME2	Throughout document
Removed content related to Error detection and reporting from the RAS Extension support chapter	-
Updated PMU events	12. Performance Monitors Extension support on page 65

Conventions

The following subsections describe conventions used in Arm documents.

Glossary

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

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	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: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Caution	We recommend the following. If you do not follow these recommendations your system might not work.
Warning	Your system requires the following. If you do not follow these requirements your system will not work.
Danger	You are at risk of causing permanent damage to your system or your equipment, or of harming yourself.
Note	This information is important and needs your attention.
Tip	This information might help you perform a task in an easier, better, or faster way.
Remember	This information reminds you of something important relating to the current content.

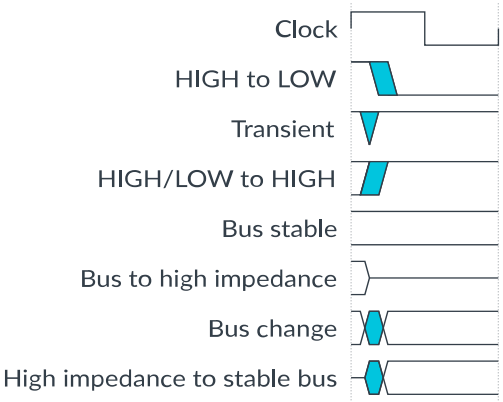
Timing diagrams

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.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 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.

Register descriptions

Reset definitions

Replication Operator {}

Verilog replication operators are used for reset values over 8-bits.

For example, `{16{1'b0}}` indicates a binary value of 16 zeros.

x

Resets that are unknown are indicated with `x`.

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Useful resources

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

Arm documents are available on developer.arm.com/documentation.

Confidential documents are only available to licensees, when logged in. Each document link in the tables below provides direct access to the online version of the document.

Arm product resources	Document ID	Confidentiality
Arm® C1-Scalable Matrix Extension 2 Configuration and Integration Manual	107832	Confidential
Arm® C1-Scalable Matrix Extension 2 Telemetry Specification	109821	Non-Confidential
Arm® C1-Scalable Matrix Extension 2 (MP207) Release Note	109578	Confidential
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual	101088	Non-Confidential
Arm® C1-DynamIQ™ Shared Unit Configuration and Integration Manual	107805	Confidential
Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual	107804	Non-Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® CHI Architecture Specification	IHI 0050	Non-Confidential
AMBA® APB Protocol Specification	IHI 0024	Non-Confidential
AMBA® ATB Protocol Specification	IHI 0032	Non-Confidential
AMBA® AXI Protocol Specification	IHI 0022	Non-Confidential
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487	Non-Confidential
Arm® Memory System Resource Partitioning and Monitoring (MPAM) System Component Specification	IHI 0099	Non-Confidential
Arm® Reliability, Availability, and Serviceability (RAS) System Architecture	IHI 0100	Non-Confidential
Arm® CoreSight™ Architecture Specification v3.0	IHI 0029	Non-Confidential

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential