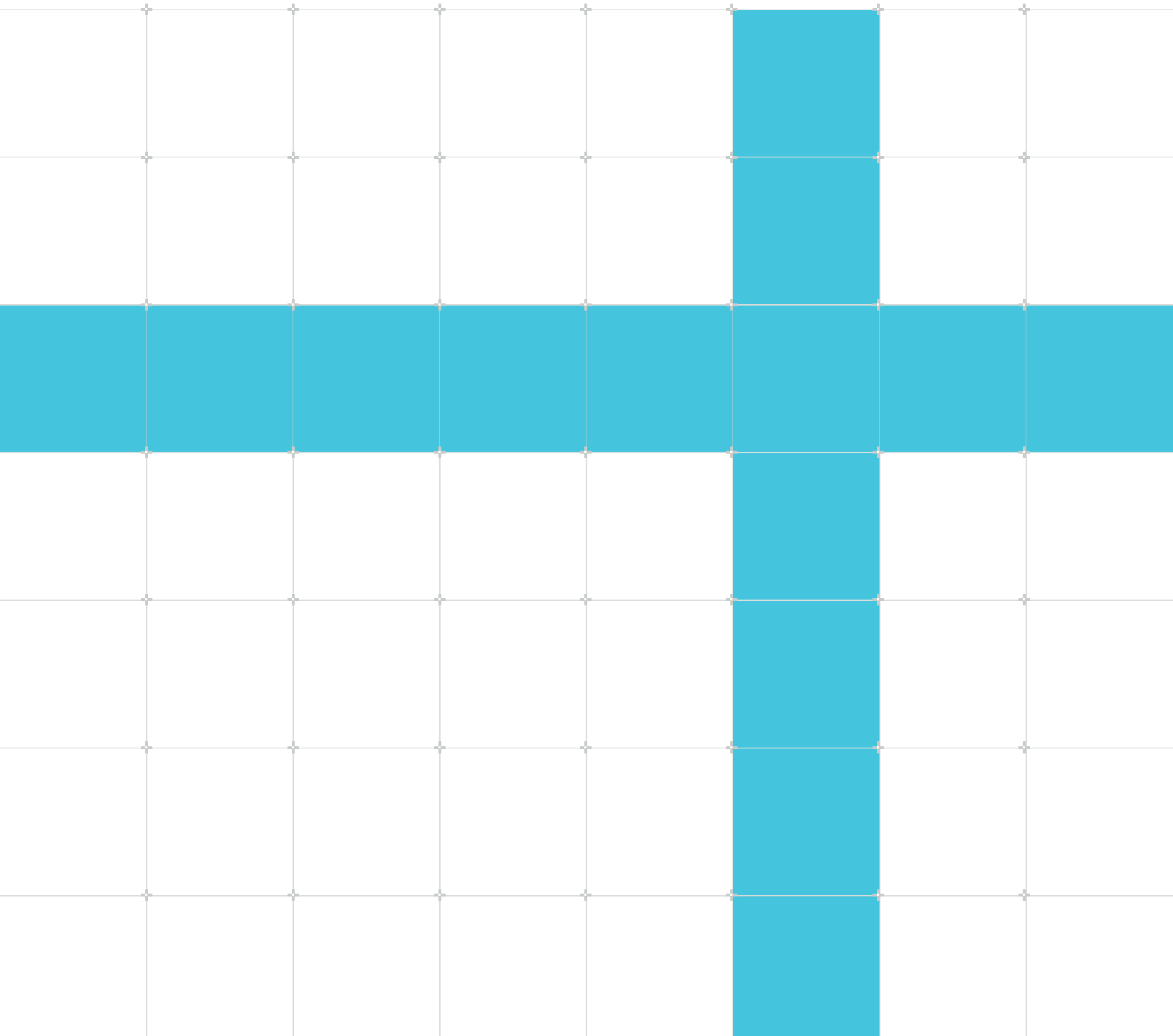
Arm® C1-Ultra Core

Revision r1p0

Technical Reference Manual

Issue 04
108014_0100_04_en



Arm® C1-Ultra Core Technical Reference Manual

This document (108014_0100_04_en) was issued on 2025-09-10. There might be a later issue at https://developer.arm.com/documentation/108014

The product revision is r1p0.

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.

We believe that this document contains no offensive language. 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

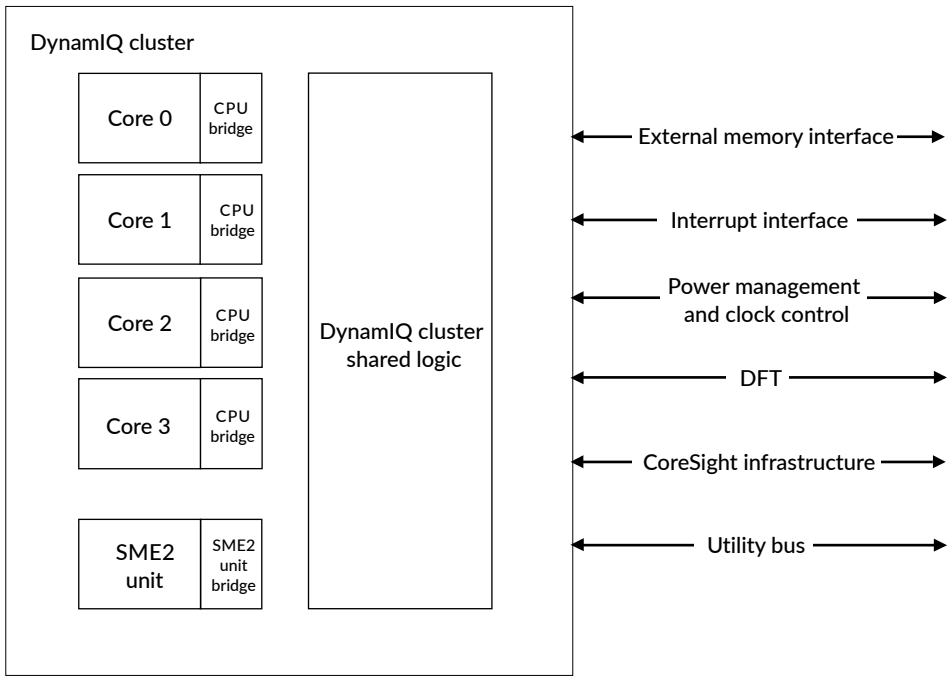
1. The C1-Ultra core

The C1-Ultra core is an ultimate-performance product that 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-Ultra core is implemented inside a C1-DSU cluster. It is connected to the C1-DynamIQ™ Shared Unit (DSU) that behaves as a full interconnect with L3 cache and snoop control. This connection configuration is also used in systems with different types of cores where the C1-Ultra core is the ultimate-performance core.

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

Figure 1-1: C1-Ultra cores example configuration



Note

- This manual applies to the C1-Ultra core only. Read this manual together with the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual* for detailed information about the C1-DSU.
- For more information on the C1-SME2 unit, see 14. Scalable Vector Extensions support on page 107 and the *Arm® C1-Scalable Matrix Extension 2 Technical Reference Manual*.

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

1.1 C1-Ultra core features

You can use the C1-Ultra core in a DynamIQ™ configuration where your homogeneous C1-DSU cluster includes one or more C1-Ultra cores. You can also use the C1-Ultra core as the ultimate-performance core in a heterogeneous cluster.

Regardless of the cluster configuration, the C1-Ultra core supports the features described in the following lists.

Core features

- Implementation of the Arm®v9.3-A A64 instruction set
- AArch64 Execution state at all Exception levels, EL0 to EL3
- Memory Management Unit (MMU)
- 40-bit Physical Address (PA) and 48-bit Virtual Address (VA)
- Generic Interrupt Controller (GIC) CPU interface to connect to an external interrupt Distributor
- Generic Timers interface that supports 64-bit count input from an external system counter
- Implementation of the Reliability, Availability, and Serviceability (RAS) Extension
- Implementation of the Scalable Vector Extension (SVE) with a 128-bit vector length and Scalable Vector Extension 2 (SVE2)
- Integrated execution unit with Advanced Single Instruction Multiple Data (SIMD) and floating-point support
- Implementation of the Scalable Matrix Extension (SME) and Scalable Matrix Extension 2 (SME2), and support for the C1-SME2 unit
- Activity Monitoring Unit (AMU)
- Support for the optional Cryptographic Extension


Note

The Cryptographic Extension is licensed separately.

Cache features

- Separate L1 data and instruction caches
- Private, unified data and instruction L2 cache

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

Debug features

- Arm®v8.8 debug architecture
- Performance Monitoring Unit (PMU)
- Embedded Trace Extension (ETE)
- TRace Buffer Extension (TRBE)
- Statistical Profiling Extension (SPE)
- Optional Embedded Logic Analyzer (ELA), ELA-600

Note

The ELA-600 is licensed separately.

Related information

2. Technical overview on page 36

1.2 C1-Ultra core configuration options

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

The C1-Ultra core implementation options include:

Cryptographic extension

You can configure your implementation with or without the Cryptographic Extension. The selected option applies to all cores in the cluster. The Cryptographic Extension is licensed separately.

L2 data RAM ECC granule

You can configure the L2 data RAM Error Correcting Code (ECC) granule to be 128 bits or 256 bits.

L2 cache size

You can configure the L2 cache to be 2048KB or 3072KB. The cores in the cluster can have different cache sizes.

PMU event counters

You can configure the number of Performance Monitoring Unit (PMU) event counters to be 6 or 31.

CoreSight™ ELA-600

You can include support for integrating CoreSight™ Embedded Logic Analyzer (ELA). This option can be configured on a per-core basis. The ELA-600 is licensed separately.

Size of the ATB FIFO depth in the core ELA

You can configure the size of the AMBA® Trace Bus (ATB) First In First Out (FIFO) to be 4, 8, 16, 32, or 64. This option can be configured on a per-core basis.

Timing closure

You can configure the L2 data cache RAMs timing behavior. For more information, see *C1-Ultra configuration parameters* in the *Arm® C1-Ultra Core Configuration and Integration Manual*.

Note

The C1-Ultra cores in the C1-DSU cluster must have identical configurations, except for the L2 cache size, ELA implementation status, and the size of the ATB FIFO depth in the core ELA.

For detailed configuration options and guidelines, see *RTL configuration process* in the *Arm® C1-Ultra Core Configuration and Integration Manual*.

1.3 C1-DSU dependent features

Some C1-DynamIQ™ Shared Unit (DSU) features and behaviors depend on whether your licensed core supports a particular feature.

The following table describes which C1-DSU dependent features are supported in your C1-Ultra core.

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

Feature	Supported in the C1-Ultra core	Dependency on the C1-DSU
Direct connect	No	-
Core included in a complex	No	-
Cryptographic Extension	Yes, as an option	Affects the external signals of the C1-DSU.
Maximum Power Mitigation Mechanism (MPMM)	Yes	For more information on MPMM, PDP, and the dispatch block signal, see 4.5 Performance and power management on page 51.
Performance Defined Power (PDP) feature	Yes	
DISPBLKy Dispatch block signal	Yes	

Feature	Supported in the C1-Ultra core	Dependency on the C1-DSU
Statistical Profiling Extension (SPE) architecture	Yes	
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*
CHI Interface Protection/ CHI DATACHECK	No	-
Scalable Matrix Extension (SME) SME2 C1-SME2	Yes	The C1-SME2 unit is required when configuring your C1-Ultra core. For more information, see the *Arm® C1-Scalable Matrix Extension 2 Technical Reference Manual*.

Note

The Cryptographic Extension is supplied under a separate license.

1.4 Supported standards, specifications, and features

The C1-Ultra core complies with the Arm®v9.3-A architecture and all previous Arm®v8-A architectures up to Arm®v8.8-A.

Supported standards and specifications

The C1-Ultra core also implements specific Arm®v8-A architecture extensions and supports interconnect, interrupt, timer, debug, and trace architectures. These extensions and architectures are documented in architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. This *Arm® C1-Ultra Core Technical Reference Manual* does not duplicate information from those sources.

Note

The C1-Ultra core is compatible with the architecture for the C1-DynamIQ™ Shared Unit (DSU) and C1-Scalable Matrix Extension 2. See the Supported standards and specifications section in both the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual* and *Arm® C1-Scalable Matrix Extension 2 Technical Reference Manual* for a list of specific architectural versions and features that the C1-DSU and C1-SME2 support.

The C1-Ultra core complies with the architectures listed in the following table.

Table 1-2: Standards and specifications supported in the C1-Ultra core

Standard or specification	Version	Notes
Arm architecture	Arm®v9.3-A	The C1-Ultra core complies with the Arm®v9.3-A architecture and all previous Arm®v8-A architectures up to Arm®v8.8-A. **Note:** The C1-Ultra core supports AArch64 only at all Exception levels, EL0 to EL3.
CoreSight™ architecture	v3.0	For more information on CoreSight™ architecture, see the *Arm® CoreSight™ Architecture Specification v3.0*.
Cryptographic Extension	Arm®v8.0-A and Arm®v8.2-A	For more information and additional cryptographic register descriptions, see the *Arm® C1-Ultra Core Cryptographic Extension Technical Reference Manual*. This extension is licensed separately and access to the documentation is restricted by contract with Arm.
Debug	Arm®v8.8	The C1-Ultra core complies with the Arm®v9.3-A architecture and is implemented with Arm®v8.8 Debug architecture support and Arm®v8.3-A Debug over PowerDown (FEAT_DoPD) support. See FEAT_DoPD in the *Arm® Architecture Reference Manual for A-profile architecture* for information on this architectural feature.
Generic Interrupt Controller (GIC) architecture CPU interface and Stream Protocol interface	GICv4.2	The C1-Ultra core uses Affinity level 1 to distinguish between different cores within the cluster. This level is not supported by some interrupt controllers, such as GIC-500. For more information about GIC architecture, see *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

Supported features

Not all architectural features are implemented in the C1-Ultra core. For a list of the architectural features implemented or not implemented in the C1-Ultra core, see the tables listed below.

Table 1-3: Implementation status of the Arm®v8.0-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_AA64	Yes	PE uses AArch64 after last reboot
FEAT_AA64EL0	Yes	Support for AArch64 at EL0
FEAT_AA64EL1	Yes	Support for AArch64 at EL1
FEAT_AA64EL2	Yes	Support for AArch64 at EL2

Feature	Implemented	Description
FEAT_AA64EL3	Yes	Support for AArch64 at EL3
FEAT_AdvSIMD	Yes	Advanced Single Instruction Multiple Data (SIMD) Extension For more information and register descriptions, see 13. Advanced SIMD and floating-point support on page 106.
FEAT_AES	Yes, configurable	Advanced SIMD Advanced Encryption Standard (AES) instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
FEAT_ASID16	Yes	16 bit ASID
FEAT_BigEnd	Yes	Big-endian support
FEAT_BigEndEL0	Yes	Big-endian support at EL0
FEAT_CLRBHB	Yes	Support for Clear Branch History instruction
FEAT_CHK	Yes	Check Feature Status
FEAT_CP15SDISABLE2	No	CP15DISABLE2
FEAT_CRC32	Yes	CRC32 instructions
FEAT_Crypto	Yes, configurable	Cryptographic Extension
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 2.1
FEAT_CSV2_3	Yes	Cache Speculation Variant 2 version 3
FEAT_CSV3	Yes	Cache Speculation Variant 3
FEAT_DGH	Yes	Data Gathering Hint
FEAT_DoubleLock	No	Double Lock
FEAT_ECBHB	Yes	Exploitative control using branch history information
FEAT_EL0	Yes	Support for execution at EL0
FEAT_EL1	Yes	Support for execution at EL1
FEAT_EL2	Yes	Support for execution at EL2
FEAT_EL3	Yes	Support for EL3
FEAT_ETS2	Yes	Enhanced Translation Synchronization
FEAT_FP	Yes	Floating Point (FP) Extension
FEAT_IVIPT	Yes	The IVIPT Extension
FEAT_LittleEnd	Yes	Little-endian support
FEAT_LittleEndEL0	Yes	Little-endian support at EL0
FEAT_MixedEnd	Yes	Mixed-endian support
FEAT_MixedEndEL0	Yes	Mixed-endian support at EL0
FEAT_nTLBPA	Yes	Intermediate caching of translation table walks
FEAT_PCSRv8	No	PC Sample-based Profiling Extension

Feature	Implemented	Description
FEAT_PMULL	Yes, configurable	Advanced SIMD PMULL instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
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_S2TGran4K	Yes	Support for 4KB memory translation granule size at stage 2
FEAT_S2TGran16K	Yes	Support for 16KB memory translation granule size at stage 2
FEAT_S2TGran64K	Yes	Support for 64KB memory translation granule size at stage 2
FEAT_SB	Yes	Speculation barrier
FEAT_Secure	Yes	Support for Secure state
FEAT_SHA1	Yes, configurable	Advanced SIMD SHA1 instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
FEAT_SHA256	Yes, configurable	Advanced SIMD SHA256 instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
FEAT_SPECRES	Yes	Speculation restriction instructions
FEAT_SPECRES2	Yes	New speculation restriction instruction
FEAT_SSBS	Yes	Speculative Store Bypass Safe (SSBS)
FEAT_SSBS2	Yes	MRS and MSR instructions for SSBS version 2
FEAT_TGran4K	Yes	Support for 4KB memory translation granule size at stage 1
FEAT_TGran16K	Yes	Support for 16KB memory translation granule size at stage 1
FEAT_TGran64K	Yes	Support for 64KB memory translation granule size at stage 1
FEAT_TRC_EXT	Yes	Trace external registers
FEAT_TRC_SR	Yes	Trace System registers

Table 1-4: Implementation status of the Arm®v8.1-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_Debugv8p1	Yes	Debug with VHE
FEAT_E2H0	Yes	Programming of HCR_EL2.E2H
FEAT_HAFDBS	Yes	Hardware management of the Access flag and dirty state
FEAT_HPDS	Yes	Hierarchical permission disables in translation tables
FEAT_LOR	Yes	Limited ordering regions
FEAT_LSE	Yes	Large System Extensions
FEAT_PAN	Yes	Privileged access never

Feature	Implemented	Description
FEAT_PAN3	Yes	Support for SCTLR_ELx.EPAN
FEAT_PMUv3p1	Yes	Armv8.1 PMU extensions
FEAT_RDM	Yes	Advanced SIMD rounding double multiply accumulate instructions
FEAT_VHE	Yes	Virtualization Host Extensions
FEAT_VMID16	Yes	16-bit VMID

Table 1-5: Implementation status of the Arm®v8.2-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_AA32BF16	No	AArch32 BFloat16 instructions
FEAT_AA32HPD	No	AArch32 Hierarchical permission disables
FEAT_AA32I8MM	No	AArch32 Int8 matrix multiplication instructions
FEAT_BF16	Yes	AArch64 BFloat16 instructions
FEAT_Debugv8p2	Yes	Debug v8.2
FEAT_DotProd	Yes	Advanced SIMD dot product instructions
FEAT_DPB	Yes	DC CVAP instruction
FEAT_DPB2	Yes	DC CVADP instruction
FEAT_EVT	Yes	Enhanced Virtualization Traps
FEAT_F32MM	No	Single-precision Matrix Multiplication
FEAT_F64MM	No	Single-precision Matrix Multiplication
FEAT_FHM	Yes	Floating-point half-precision multiplication instructions
FEAT_FP16	Yes	Half-precision floating-point data processing
FEAT_HPDS2	Yes	Hierarchical permission disables
FEAT_I8MM	Yes	AArch64 Int8 matrix multiplication instructions
FEAT_IESB	Yes	Implicit Error Synchronization event
FEAT_LPA	No	Large PA and IPA support
FEAT_LSMAOC	No	AArch32 Load/Store Multiple instruction atomicity and ordering controls
FEAT_LVA	No	Large VA support
FEAT_MPAM	Yes	Memory Partitioning and Monitoring (MPAM) Extension For more information on the MPAM Extension, see the *Arm® Memory System Resource Partitioning and Monitoring (MPAM) System Component Specification* and the *Arm® Architecture Reference Manual for A-profile architecture*.
FEAT_MPAMv1p0	Yes	Memory Partitioning and Monitoring Extension
FEAT_PAN2	Yes	AT S1E1R and AT S1E1W instruction variants affected by PSTATE.PAN
FEAT_PCSRv8p2	Yes	PC Sample-based Profiling Extension
FEAT_RAS	Yes	Reliability, Availability, and Serviceability (RAS) Extension
FEAT_RASSA	Yes	RAS System Architecture
FEAT_RASSAv1	Yes	RAS version 1 System Architecture

Feature	Implemented	Description
FEAT_SHA3	Yes, configurable	Advanced SIMD SHA3 instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
FEAT_SHA512	Yes, configurable	Advanced SIMD SHA512 instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
FEAT_SM3	Yes, configurable	Advanced SIMD SM3 instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
FEAT_SM4	Yes, configurable	Advanced SIMD SM4 instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
FEAT_SPE	Yes	Statistical Profiling Extension (SPE) For more information, see 21. Statistical Profiling Extension support on page 198.
FEAT_SPE_LDS	Yes	Statistical Profiling data source packet generation
FEAT_SpecSEI	No	SError interrupt exceptions from speculative reads of memory
FEAT_SVE	Yes	Scalable Vector Extension (SVE)
FEAT_TTCNP	Yes	Translation table Common not private translations
FEAT_UAO	Yes	Unprivileged Access Override control
FEAT_VPIPT	No	VMID-aware PIPT instruction cache
FEAT_XNX	Yes	Translation table stage 2 Unprivileged Execute-never

Table 1-6: Implementation status of the Arm®v8.3-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_CCIDX	Yes	Extended cache index
FEAT_CONSTPACFIELD	Yes	PAC algorithm enhancement
FEAT_DoPD	Yes	Debug over Powerdown
FEAT_EPAC	No	Enhanced pointer authentication
FEAT_FCMA	Yes	Floating-point complex number instructions
FEAT_FPAC	Yes	Faulting on AUT* instructions
FEAT_FPACC_SPEC	Yes	Faulting on combined pointer authentication instructions
FEAT_FPACCOMBINE	Yes	Faulting on combined pointer authentication instructions
FEAT_JSCVT	Yes	JavaScript conversion instruction
FEAT_LRCPC	Yes	Load-Acquire RCpc instructions

Feature	Implemented	Description
FEAT_NV	No	Nested Virtualization
FEAT_PACIMP	No	Pointer authentication - IMPLEMENTATION DEFINED algorithm
FEAT_PACQARMA3	Yes	Pointer authentication - QARMA3 algorithm
FEAT_PACQARMA5	No	Pointer authentication - QARMA5 algorithm
FEAT_PAuth	Yes	Pointer authentication
FEAT_PAuth2	Yes	Enhancements to pointer authentication
FEAT_SPEv1p1	Yes	Statistical Profiling Extensions version 1.1

Table 1-7: Implementation status of the Arm®v8.4-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_AMU_EXT	Yes	External Activity Monitors
FEAT_AMU_EXT32	Yes	32-bit External Activity Monitors extension
FEAT_AMUv1	Yes	Activity Monitors Extension version 1
FEAT_BBML1	Yes	Translation table break-before-make levels
FEAT_BBML2	Yes	Translation table break-before-make levels
FEAT_Debugv8p4	Yes	Debug v8.4
FEAT_DIT	Yes	Data Independent Timing instructions
FEAT_DoubleFault	Yes	Double Fault Extension
FEAT_FlagM	Yes	Condition flag manipulation instructions
FEAT_IDST	Yes	ID space trap handling
FEAT_LRCPC2	Yes	Load-Acquire RCpc instructions version 2
FEAT_LSE2	Yes	Large System Extensions version 2
FEAT_NV2	No	Enhanced nested virtualization support
FEAT_PMUv3p4	Yes	Armv8.4 PMU Extensions
FEAT_RASSAv1p1	Yes	RAS version v1.1 System Architecture
FEAT_RASv1p1	Yes	Reliability, Availability, and Serviceability (RAS) Extension v1.1 All extensions up to Arm®v9.0-A at full containment capability with Error Correcting Code (ECC) configured. For more information on the implementation of the RAS Extension, see 10. RAS extension support on page 94.
FEAT_S2FWB	Yes	Stage 2 forced Write-Back
FEAT_SEL2	Yes	Secure EL2
FEAT_TLBIOS	Yes	TLB invalidate instructions in Outer Shareable domain
FEAT_TLBIRANGE	Yes	TLB invalidate range instructions
FEAT_TRF	Yes	Self-hosted Trace extensions
FEAT_TTL	Yes	Translation Table Level
FEAT_TTST	Yes	Small translation tables

Table 1-8: Implementation status of the Arm®v8.5-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_BTI	Yes	Branch Target Identification (BTI)
FEAT_E0PD	Yes	Preventing EL0 access to halves of address maps
FEAT_ExS	No	Disabling context synchronizing exception entry and exit
FEAT_FlagM2	Yes	Condition flag manipulation version 2
FEAT_FRINTTS	Yes	FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions
FEAT_GTG	Yes	Guest translation granule size
FEAT_MTE	Yes	Instruction-only Memory Tagging Extension (MTE) **Note:** The C1-Ultra core always implements 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_MTE_ASYM_FAULT	Yes, configurable	Memory tagging asymmetric faults
FEAT_MTE_ASYNC	Yes, configurable	Memory Tagging asynchronous faulting
FEAT_MTE2	Yes, configurable	Memory Tagging Extension version 2
FEAT_MTE3	Yes, configurable	MTE Asymmetric Fault Handling
FEAT_PMUv3p5	Yes	Armv8.5 PMU Extensions
FEAT_RNG	No	Random number generator
FEAT_RNG_TRAP	Yes	Trapping support for RNDR/RNDRRS

Table 1-9: Implementation status of the Arm®v8.6-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_AMUv1p1	No	AMU Extensions version 1.1
FEAT_ECV	Yes	Enhanced Counter Virtualization
FEAT_ECV_POFF	Yes	Enhanced Counter Virtualization
FEAT_FGT	Yes	Fine Grain Traps
FEAT_MPAMv0p1	No	Memory Partitioning and Monitoring version 0.1
FEAT_MPAMv1p1	Yes	Memory Partitioning and Monitoring version 1.1
FEAT_MTPMU	No	Multi-threaded PMU Extensions
FEAT_TWED	No	Delayed Trapping of WFE

Table 1-10: Implementation status of the Arm®v8.7-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_AFP	Yes	Alternate floating-point behavior
FEAT_HCX	Yes	Support for the HCRX_EL2 register
FEAT_LPA2	No	Larger physical address for 4KB and 16KB translation granules
FEAT_LS64	No	Support for 64 byte loads and stores without return
FEAT_LS64_ACCDATA	No	Support for 64-byte EL0 stores with return

Feature	Implemented	Description
FEAT_LS64_V	No	Support for 64-byte stores with return
FEAT_PMUv3p7	Yes	Armv8.7 PMU Extensions For more information on PMU Extensions, see 17. Performance Monitors Extension support on page 125.
FEAT_RPRES	No	Increased precision of Reciprocal Estimate and Reciprocal Square Root Estimate
FEAT_SPE_FnE	Yes	Statistical Profiling inverse event filter
FEAT_SPEv1p2	Yes	Statistical Profiling Extensions version 1.2 For more information on PMU Extensions, see 21. Statistical Profiling Extension support on page 198.
FEAT_WFxT	Yes	WFE and WFI instructions with timeout For more information on PMU Extensions, see 4.2.1 Wait for Interrupt and Wait for Event on page 45.
FEAT_XS	Yes	XS attribute

Table 1-11: Implementation status of the Arm®v8.8-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_CMOW	Yes	Control for cache maintenance permission
FEAT_Debugv8p8	Yes	Debug v8.8
FEAT_HBC	Yes	Hinted conditional branch
FEAT_HPMN0	Yes	Setting of MDCR_EL2.HPMN to zero
FEAT_MOPS	Yes	Standardization of memory operations
FEAT_NMI	Yes	Non-maskable Interrupts
FEAT_PMUv3_TH	No	Event counting threshold
FEAT_PMUv3p8	Yes	Armv8.8 PMU extensions
FEAT_SCTLR2	No	Extension to SCTLE_ELx
FEAT_SPEv1p3	Yes	Statistical Profiling Extension version 1.3
FEAT_TCR2	Yes	Extension to TCR_ELx
FEAT_TIDCP1	Yes	EL0 use of IMPLEMENTATION DEFINED functionality

Table 1-12: Implementation status of the Arm®v8.9-A features in the C1-Ultra core

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

Feature	Implemented	Description
FEAT_EDHSR	Yes	Support for External Debug Halt Status Register (EDHSR)
FEAT_FGT2	No	Fine-grained traps 2
FEAT_HAFT	Yes	Hardware-managed Access Flag for Table descriptors
FEAT_LRCPC3	Yes	Load-Acquire RCpc instructions version 3
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_PERM	Yes, configurable	Allocation tag access permission
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, configurable	Support for Range Prefetch Memory instruction
FEAT_S1PIE	No	Permission model enhancements
FEAT_S1POE	No	Permission model enhancements
FEAT_S2PIE	No	Permission model enhancements
FEAT_S2POE	No	Permission model enhancements
FEAT_SPE_CRR	No	Call Return Branch Records
FEAT_SPE_DPFZS	Yes	Disable Cycle Counter on SPE Freeze
FEAT_SPE_FDS	No	Data Source Filtering
FEAT_SPEv1p4	No	Statistical Profiling Extension version 1.4
FEAT_SPMU	No	System Performance Monitors Extension
FEAT_THE	No	Translation Hardening Extension

Table 1-13: Implementation status of the Arm®v9.0-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_Armv9_Crypto	Yes, configurable	Armv9 Cryptographic Extension
FEAT_ETE	Yes	Embedded Trace Extension (ETE) For more information on ETE, see 18. Embedded Trace Extension support on page 180.

Feature	Implemented	Description
FEAT_SVE_AES	Yes, configurable	Scalable Vector Extension (SVE) Advanced Encryption Standard (AES) instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
FEAT_SVE_BitPerm	Yes	SVE Bit Permutes instructions
FEAT_SVE_PMULL128	Yes, configurable	SVE PMULL instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
FEAT_SVE_SHA3	Yes, configurable	SVE SHA3 instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
FEAT_SVE_SM4	Yes, configurable	SVE SM4 instructions **Note:** Supported as part of the Arm®v8-A Cryptographic Extension
FEAT_SVE2	Yes	SVE version 2 For more information on SVE version 2, see 14. Scalable Vector Extensions support on page 107.
FEAT_TME	No	Transactional Memory Extension (TME)
FEAT_TRBE	Yes	Trace Buffer Extension (TRBE) For more information on TRBE, see 19. Trace Buffer Extension support on page 191.

Table 1-14: Implementation status of the Arm®v9.1-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_ETEv1p1	Yes	Embedded Trace Extension (ETE) version 1.1

Table 1-15: Implementation status of the Arm®v9.2-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_BRBE	No	Branch Record Buffer Extension
FEAT_EBF16	No	AArch64 Extended BFloat16 instructions
FEAT_ETEv1p2	No	Embedded Trace Extension (ETE) version 1.2
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

Feature	Implemented	Description
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_RME	No	Realm Management Extension (RME)
FEAT_SME	Yes	Scalable Matrix Extension (SME)
FEAT_SME_F64F64	No	Double-precision FP outer product instructions
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_FA64	No	Full A64 instruction set support in Streaming Scalable Vector Extension (SVE) mode
FEAT_SME_I16I64	No	16-bit to 64-bit integer widening outer product instructions
FEAT_SME_LUTv2	No	Lookup table instructions with 4-bit indices and 8-bit elements
FEAT_SPE_SME	Yes	Statistical Profiling extensions for SME
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

Table 1-16: Implementation status of the Arm®v9.3-A features in the C1-Ultra core

Feature	Implemented	Description
FEAT_BRBEv1p1	No	Branch Record Buffer Extension version 1.1
FEAT_MEC	No	Memory Encryption Contexts (MEC)
FEAT_SME2	Yes	Scalable Matrix Extension (SME) version 2

Table 1-17: Arm®v9.4-A features implemented in the C1-Ultra core

Feature	Implemented	Description
FEAT_ABLE	No	Address Breakpoint Linking Extension
FEAT_BWE	No	Breakpoint and watchpoint enhancements
FEAT_D128	No	128-bit Translation Tables, 56 bit PA
FEAT_EBEP	No	Exception-based event profiling
FEAT_ETEv1p3	No	Embedded Trace Extension version 1.3
FEAT_GCS	No	Guarded Control Stack Extension
FEAT_ITE	No	Instrumentation Trace Extension
FEAT_LSE128	No	128-bit Atomics
FEAT_LVA3	No	56-bit VA
FEAT_SVE2p1	No	Scalable Vector Extensions version 2.1

Feature	Implemented	Description
FEAT_SEBEP	No	Synchronous Exception-based Event Profiling
FEAT_SME_F16F16	No	Non-widening half-precision FP16 to FP16 arithmetic for SME2
FEAT_SME2p1	No	Scalable Matrix Extension version 2.1
FEAT_SVE_B16B16	No	Non-widening BFloat16 to BFloat16 arithmetic for SVE2 and SME2
FEAT_SYSINSTR128	No	128-bit System instructions
FEAT_SYSREG128	No	128-bit System registers
FEAT_TRBE_EXT	No	Trace Buffer external mode
FEAT_TRBE_MPAM	No	Trace Buffer MPAM extensions

GIC supported features

The following table shows the Generic Interrupt Controller (GIC) features and whether they are implemented or not implemented in the C1-Ultra core.

Table 1-18: Implementation status of the GIC features in the C1-Ultra core

Feature	Implemented	Description
FEAT_GICv3	Yes, configurable	GIC version 3
FEAT_GICv3_LEGACY	No	Support for GICv2 legacy operations
FEAT_GICv3_NMI	Yes, configurable	GIC Non-maskable Interrupts
FEAT_GICv3_TDIR	Yes, configurable	Trapping Non-secure EL1 writes to ICV_DIR
FEAT_GICv3p1	Yes, configurable	GIC version 3.1
FEAT_GICv4	Yes, configurable	GIC version 4
FEAT_GICv4p1	Yes, configurable	GIC version 4.1

For more information on the GIC architecture, see *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

Related information

2.1 Core components on page 36

1.5 Test features

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

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

Arm also provides MBIST interfaces that enable you to test the RAMs at operational frequency. You can add your own MBIST controllers to automatically generate test patterns and perform result comparisons. Optionally, you can use your Electronic Design Automation (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-Ultra Core 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-Ultra Core 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-Ultra core is delivered as an RTL description in SystemVerilog that can be synthesized. Before you can use the C1-Ultra core, you must implement, integrate, and program it.

A different group can perform each of the following tasks:

Implementation

The implementer configures the RTL, adds vendor cells or 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.

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

Software programming

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

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

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

See *RTL configuration process* in the *Arm® C1-Ultra Core Configuration and Integration Manual* and in the *Arm® C1-DynamIQ™ Shared Unit Configuration and Integration Manual* for implementation options. See also *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 1-19: Product revisions

Revision	Notes
r0p0	First release
r1p0	First release

Changes in functionality that have an impact on the documentation also appear in Revision history on page 897.

2. Technical overview

The C1-Ultra core 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 main blocks include:

- L1 instruction and L1 data memory systems
- L2 memory system
- Register rename
- Instruction decode
- Instruction issue
- Execution pipeline
- Memory Management Unit (MMU)
- Trace unit and trace buffer
- Performance Monitoring Unit (PMU)
- Activity Monitoring Unit (AMU)
- Generic Interrupt Controller (GIC) CPU interface
- Branch prediction

The programmer's model and the architecture features implemented, such as the Generic Timer, are compliant with the standards in 1.4 Supported standards, specifications, and features on page 21.

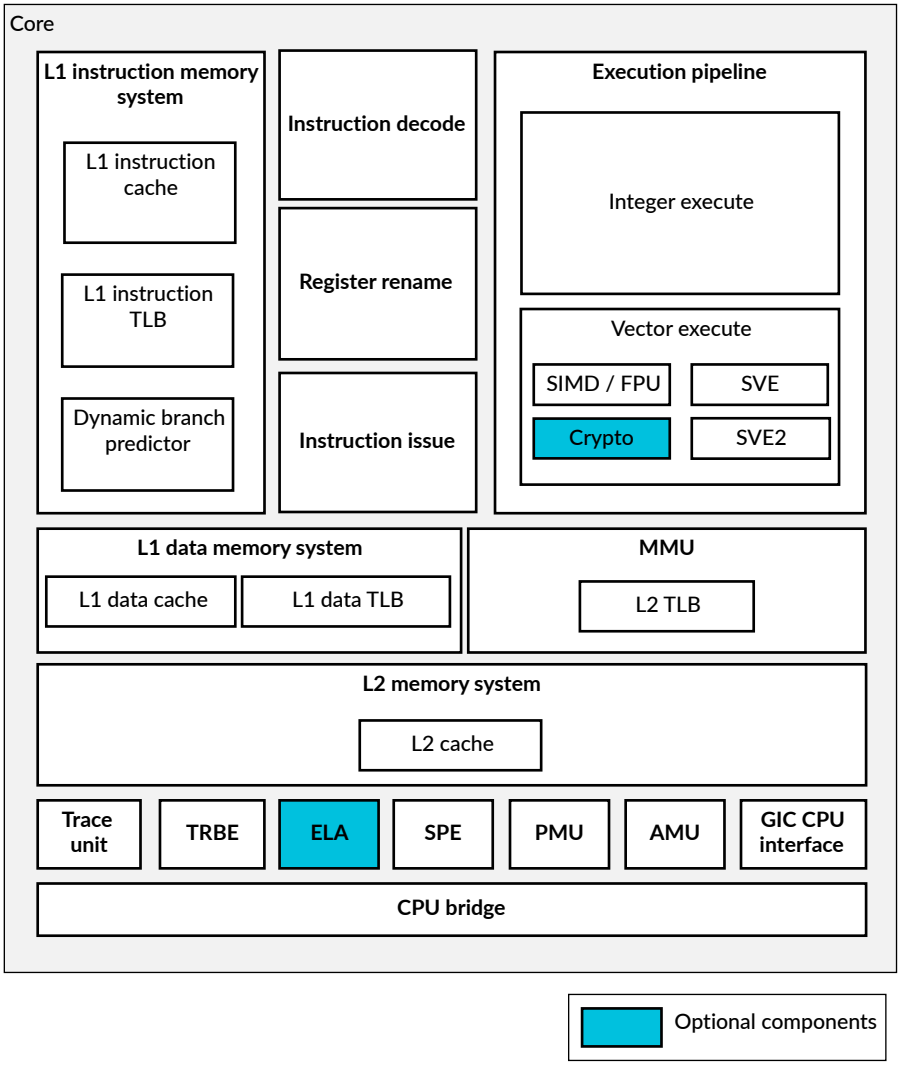
The C1-Ultra core interfaces with the C1-DynamIQ™ Shared Unit (DSU) through the CPU bridge.

2.1 Core components

The C1-Ultra core includes components designed to make it an ultimate-performance product. The C1-Ultra core includes a CPU bridge that connects the core to the C1-DynamIQ™ Shared Unit (DSU). The C1-DSU connects the core to an external memory system and the rest of the System on Chip (SoC).

The following figure shows the C1-Ultra core components.

Figure 2-1: C1-Ultra core components



L1 instruction memory system

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

The L1 instruction memory system includes:

- A 64KB, 4-way set associative L1 instruction cache with 64-byte cache lines.
- A fully associative L1 instruction Translation Lookaside Buffer (TLB) with native support for 4KB, 16KB, 64KB, and 2MB page sizes.
- A dynamic branch predictor.

Instruction decode

The instruction decode unit decodes AArch64 instructions into internal format.

Register rename

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

Instruction issue

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

Integer execute

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

Vector execute

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

Advanced SIMD and floating-point support

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

Cryptographic Extension

The Cryptographic Extension is optional in the C1-Ultra core. The Cryptographic Extension adds new instructions to the Advanced SIMD and the Scalable Vector Extension (SVE) instruction sets that accelerate:

- Advanced Encryption Standard (AES) encryption and decryption
- The Secure Hash Algorithm (SHA) functions SHA1, SHA2, SHA3
  - The SVE2 versions of the SHA3 instructions EOR3, XAR, and BCAX are supported even when CRYPTO support is not configured.
- Armv8.2-SM SM3 hash function and SM4 encryption and decryption instructions
- Finite field arithmetic that is used in algorithms such as Galois/Counter Mode and Elliptic Curve Cryptography


Note

The optional Cryptographic Extension is not included in the base product. Arm supplies the Cryptographic Extension under an additional license to the C1-Ultra core license.

Scalable Vector Extension

The Scalable Vector Extension (SVE) and Scalable Vector Extension 2 (SVE2) are extensions to the Armv8-A architecture.

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


Note

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

L1 data memory system

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

The L1 data memory system includes:

- A 128KB, 4-way set associative cache with 64-byte cache lines.
- A fully associative L1 data TLB with native support for 4KB, 16KB, and 64KB page sizes and 2MB and 512MB block sizes.

Memory Management Unit

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

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

L2 memory system

The L2 memory system includes the L2 cache. The L2 cache is private to the core and can be configured to be 2MB 8-way set associative or 3MB 12-way set associative. The L2 memory system is connected to the C1-DSU through an asynchronous CPU bridge.

Embedded Trace Extension and Trace Buffer Extension

The C1-Ultra core supports a range of debug, test, and trace options including a trace unit and a trace buffer.

The C1-Ultra core also includes a ROM table that contains a list of components in the system. Debuggers can use the ROM table to determine which CoreSight components are implemented.

All the debug and trace components of the C1-Ultra core are described in this manual. For more information about the Embedded Logic Analyzer (ELA), see the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual*.

Statistical Profiling Extension

The Statistical Profiling Extension (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 or 31 performance monitors, depending on your configuration. The performance monitors can be configured to gather statistics on the operation of each core and the memory system. The information can be used for debug and code profiling.

Activity Monitoring Unit

Activity monitors in the Activity Monitoring Unit (AMU) provide useful information for system power management and persistent monitoring.

GIC CPU interface

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

CPU bridge

In a cluster, there is one CPU bridge between each C1-Ultra core and the C1-DSU.

The CPU bridge controls buffering and synchronization between the core and the C1-DSU.

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

Related information

5. Memory management on page 56
6. L1 instruction memory system on page 65
7. L1 data memory system on page 69
8. L2 memory system on page 74
12. GIC CPU interface on page 102
13. Advanced SIMD and floating-point support on page 106
17. Performance Monitors Extension support on page 125
18. Embedded Trace Extension support on page 180

2.2 Interfaces

The C1-DynamIQ™ Shared Unit (DSU) manages all C1-Ultra core external interfaces to the System on Chip (SoC).

See the *Technical overview* chapter in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual* for detailed information on these interfaces.

2.3 Programmer's model

The C1-Ultra core 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-Ultra core supports the AArch64 Execution state at all Exception levels, EL0 to EL3.

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

Related information

1.4 Supported standards, specifications, and features on page 21

3. Clocks and resets

To provide dynamic power savings, the C1-Ultra core supports hierarchical clock gating. It also supports Warm and Cold resets.

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

In addition, the C1-Ultra core implements extensive clock gating that includes:

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

The C1-Ultra core receives the following reset signals from the C1-DynamIQ™ Shared Unit (DSU) side of the CPU bridge:

- A Warm reset for all registers in the core except for:
  - Some parts of the debug logic
  - Some parts of the trace logic
  - Reliability, Availability, and Serviceability (RAS) logic
  - Performance and Power Management registers. See 4.5 Performance and power management on page 51 for more details on Performance and power management registers.
- A Cold reset for the logic in the core, including the debug logic, trace logic, and RAS logic.

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

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

4. Power management

The C1-Ultra core provides mechanisms to control both dynamic and static power dissipation.

The dynamic power management includes the following features:

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

The static power management includes the following features:

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

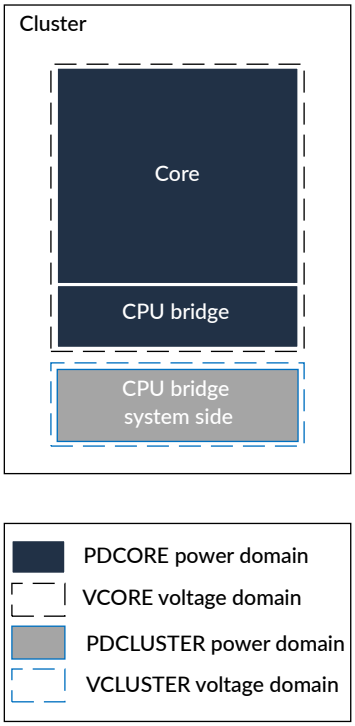
4.1 Voltage and power domains

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

The PDCORE power domain contains all C1-Ultra core logic and part of the core asynchronous bridge that belongs to the VCORE domain. The PDCLUSTER power domain contains the part of the CPU bridge that belongs to the VCLUSTER domain.

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

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



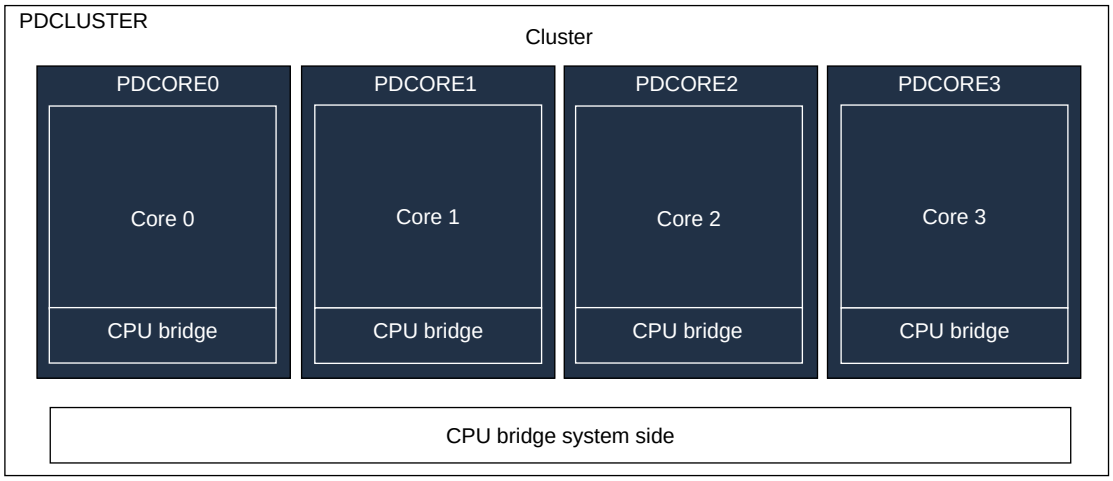
You can tie the VCORE and VCLUSTER voltage domains to the same supply if one of the following is true:

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

In a cluster with multiple C1-Ultra cores, there is one PDCORE<n> power domain per core, where n is the core instance number. If a core is not present, then the corresponding power domain is not present.

The following figure shows an example of the power domains with four C1-Ultra cores in a cluster.

Figure 4-2: Core power domains in a cluster with four C1-Ultra cores



Clamping cells between power domains are inferred through power intent files rather than instantiated in the RTL. For more information, see *Power management* in the *Arm® C1-Ultra Core Configuration and Integration Manual*.

For detailed 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

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

4.2.1 Wait for Interrupt and Wait for Event

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

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

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

Note

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

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

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

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

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

4.2.2 Low-power state behavior considerations

You must consider how certain events affect the Wait for Interrupt (WFI) and Wait for Event (WFE) low-power state behavior of the C1-Ultra core.

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

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

Note

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

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

4.3 Power control

The C1-DynamIQ™ Shared Unit (DSU) Power Policy Units (PPUs) control all core and cluster power mode transitions.

Each core has its own PPU to control its own core power domain.

In addition, there is a PPU for the cluster.

The PPUs control and request any change in power mode. The C1-Ultra core then performs any actions necessary to reach the requested power mode. For example, the core might gate clocks, clean caches, or disable coherency before it accepts the request.

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

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

4.4 Core power modes

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

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

The following table shows the supported C1-Ultra core power modes.

Power modes that are not shown in the following table are not supported and must not occur. Deviating from the legal power modes can lead to **UNPREDICTABLE** results. You must comply with the dynamic power management and powerup and powerdown sequences described in 4.6 C1-Ultra core powerup and powerdown sequence on page 53.

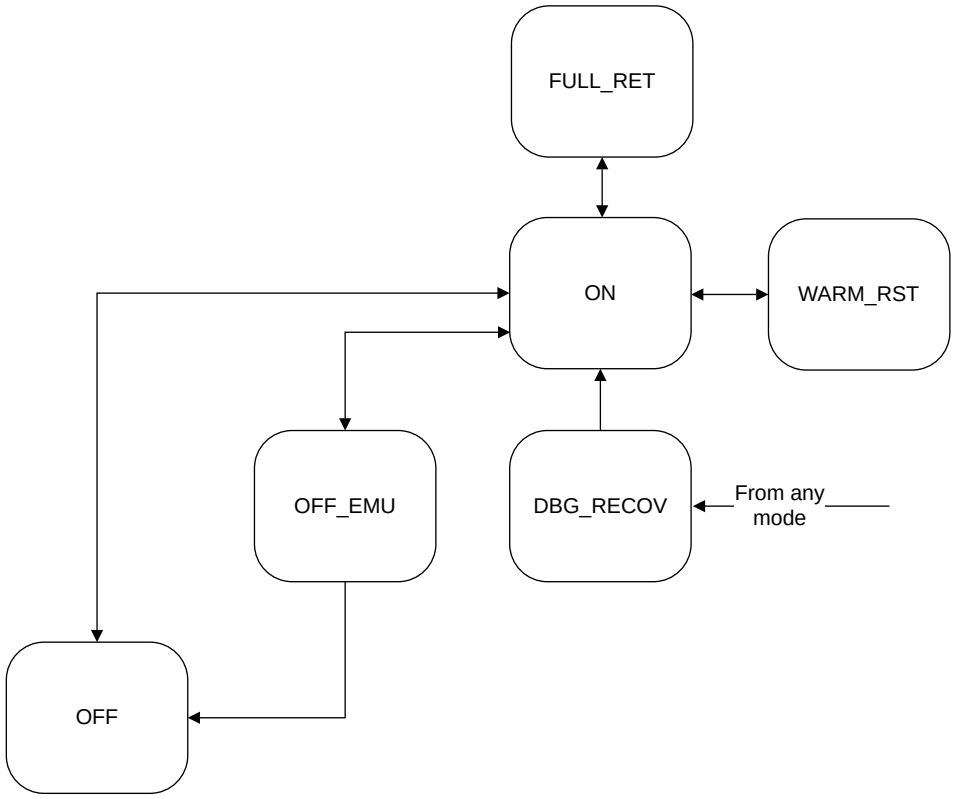
Table 4-1: C1-Ultra core power modes

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

Power mode	Short name	Power state
Debug recovery	DBG_RECOV	The RAM and logic are powered up. This mode is for applying 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 state except for the debug logic, the trace unit logic, the Activity Monitor Unit (AMU) logic, and the debug and the RAS registers.

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

Figure 4-3: C1-Ultra core power mode transitions



Related information

4.2 Architectural clock gating modes on page 45
4.2.1 Wait for Interrupt and Wait for Event on page 45
4.4.4 Full retention mode on page 49

4.4.1 On mode

In the On power mode, the C1-Ultra core is on and fully operational.

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

4.4.2 Off mode

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

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

A transition from Off mode to On mode applies a Cold reset to the core.

Attempted debug access or utility bus access to the core, when the core domain is off, returns an error response on the internal debug interface and utility bus. This error response indicates the core is unavailable. For more information, see *Utility bus* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

Note

The core-specific debug registers in the DebugBlock for External Debug Over Powerdown (EDOP) feature can be accessed while the core is in Off mode.

4.4.3 Emulated off mode

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

4.4.4 Full retention mode

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

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

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

- The core is in Wait for Interrupt (WFI) or Wait for Event (WFE) low-power state.

- The retention timer has expired. For more information on setting the retention timer, see A.4.14 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 328.
- The core clock is temporarily disabled for any reason stated in 4.2.2 Low-power state behavior considerations on page 46.

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

- A WFI or WFE wakeup event, as defined in the *Arm® Architecture Reference Manual for A-profile architecture*.
- An event that requires the core clock to be temporarily enabled without exiting the WFI or WFE low-power state. For example:
  - L1 snoops or L2 snoops
  - Cache or TLB maintenance operations
  - Debug access from the DebugBlock of the C1-DynamIQ™ Shared Unit (DSU)
  - GIC access

Related information

4.2.1 Wait for Interrupt and Wait for Event on page 45

4.4.5 Debug recovery mode

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

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

In addition to preserving the cache contents, Debug recovery mode supports preserving the Reliability, Availability, and Serviceability (RAS) state, but can only be preserved if starting from the on state. A transition to Debug recovery mode is made from any state, which puts the core 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).

For more information on the C1-DSU PPUs, see *The Power Policy Unit* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

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

Debug recovery mode can occur at any time with no guarantee of the state of the core. A request of this type is accepted immediately, therefore its effects on the core, the 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. The core is not expecting these transactions to complete after a reset, and might cause a system deadlock.

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

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

4.4.6 Warm reset mode

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

WARM_RST mode is strictly for debug purposes.

A Warm reset is applied to the C1-Ultra core when the core 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 core. A request to transition to WARM_RST mode is accepted immediately. Therefore, its effects on the core, the DynamIQ™ cluster, or the wider system are **UNPREDICTABLE**, and a wider system reset might be required. 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.

An alternative method for placing the core into Warm reset is to use the Arm®v8-A Reset Management Register, RMR_EL3. When the core runs in EL3, it requests a Warm reset of the core if you set the RMR_EL3.RR bit to 1. If RMR_EL3.RR is set to 1 before a WFI instruction is executed, then the core will request a Warm reset. The RMR_EL3.RR controlled Warm reset of the core is independent of the PPU WARM_RST power mode.
For more information about RMR_EL3, see the *Arm® Architecture Reference Manual for A-profile architecture*.

4.5 Performance and power management

The C1-Ultra core implements Performance and Power Management (PPM) features that can be used to limit high activity events within the core, or trade off efficiency versus peak performance.

The PPM features supported by the C1-Ultra core are:

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

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 high-power load-store events and vector unit instructions.

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

MPMM provides three gears that enable it to limit certain classes of workloads. Each MPMM gear limits workloads at a different level of aggressiveness, where gear 0 produces the most aggressive throttling and gear 2 the least aggressive. The Activity Monitoring Unit (AMU) provides metrics for each gear.

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

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

An external power controller can use these AMU metrics to budget System on Chip (SoC) power in the following ways:

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

Related information

A.9.2 IMP_CPUPPMCR_EL3, Global Performance and Power Management Configuration Register on page 506
A.9.3 IMP_CPUMPMMCR_EL3, Global MPMM Control Register on page 510
B.7.1 CPUPPMCR, Global Performance and Power Management Configuration Register on page 822
B.7.2 CPUMPMMCR, Global MPMM Control Register on page 826

4.5.2 Performance Defined Power

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

The PDP has an impact on:

- Core power reduction. The core power is reduced and the efficiency is increased.

4.5.3 Dispatch block

In extreme core thermal or power conditions, you can temporarily halt forward progress of the core without stopping the clock.

A pin is provided on the C1-DSU boundary that can directly be used to force the core to stall for the duration that the pin is asserted. When the core is stalled, the dispatch of new instructions is stopped. However, instructions that have already been dispatched continue to execute and complete as normal.

4.6 C1-Ultra core powerup and powerdown sequence

There is no specific sequence to power up the C1-Ultra core or bring it into coherence after reset. To power down the core, you must follow a specific sequence.

To power down the C1-Ultra core:

1. If required, save the state of the core to system memory to allow for retrieval of the core state during powerup.
2. Ensure the C1-SME2 is properly disconnected by setting PSTATE.SM and PSTATE.ZA to 0. It will not be possible to power down until the core is fully disconnected from C1-SME2.
3. Disable interrupts to the core.
    a. Disable the interrupt enable bits in the ICC_IGRPEN0_EL1 and ICC_IGPREN1_EL1 registers.
    b. Set the GIC distributor wake-up request for the core using the GICR_WAKER register.
    c. Read the GICR_WAKER register to confirm that the ChildrenAsleep bit indicates that the interface is inactive.
4. Disable the interrupt outputs from the Reliability, Availability, and Serviceability (RAS) registers or redirect the core RAS fault and error interrupt outputs to the system error manager.
5. Set the IMP_CPUPWRCTLR_EL1.CORE_PWRDN_EN bit to 1 to indicate to the power controller that a powerdown is requested.
6. Execute an `ISB` instruction.
7. Execute a `WFI` instruction. Once the `WFI` instruction is executed, the powerdown sequence cannot be interrupted except as shown in Aborts to the powerdown sequence on page 54.

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

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

Aborts to the powerdown sequence

When the IMP_CPUPWRCTLR_EL1.CORE_PWRDN_EN bit is set, executing a `WFI` instruction automatically masks interrupts and wakeup events in the core. Typically, applying a reset is the only way to wake up the core from the Wait for Interrupt (WFI) state. However, if any of the following events occur during the powerdown sequence, the core will abort the powerdown sequence and wake up from the WFI state:

A General Interrupt Controller (GIC) interrupt

A RAS fault or error interrupt

If a Reliability, Availability, and Serviceability (RAS) fault or error interrupt is signaled from the core during the powerdown sequence, then the core denies the powerdown request to ensure that information about the fault is not lost. Software or firmware must clear the interrupt source in the RAS registers before it attempts to power down again.

Alternatively, the firmware can disable the RAS fault and error interrupt outputs before executing the powerdown `WFI` instruction to prevent faults from causing powerdown denial. However, any faults or errors detected during the powerdown sequence would then not be reported. Any records of the fault or error would be lost.

A pending interrupt is waiting to be serviced

If a pending interrupt is waiting to be serviced, the core might deny the powerdown request because of the COREWAKEREQUEST signal being asserted during the powerdown. The core cannot power down while the interrupt is pending. Interrupts must be re-enabled and the interrupt serviced before re-attempting powerdown.

Transient conditions exist that affect powerdown

The core might deny the powerdown request because of the transient conditions coming from the system during the powerdown sequence. These events are rare, and if the powerdown request is repeated it is likely to succeed.

To handle these cases, the firmware that executed the `WFI` instruction must be designed to cope with execution continuing after the `WFI` instruction. It might need to restore some state, re-enable interrupts, and hand back control to the operating system. If the software has handled any RAS faults reported, and has determined that there is no other reason to remain powered on, then it can restart the powerdown sequence, including saving the core state if required and disabling interrupts.

4.6.1 Managing RAS fault and error interrupts during the core powerdown sequence

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

The Wait for Interrupt (WFI) instruction is normally the point of no return for powering down the core. However, if a Reliability, Availability, and Serviceability (RAS) fault or error interrupt is signaled from the core during the powerdown sequence. This causes the core to deny the powerdown request and causes the core to wake up from WFI.

Therefore, if the RAS fault and error interrupt outputs remain active during the powerdown sequence, the software must be designed so that if it wakes up from the powerdown WFI, it can analyze the RAS fault or error and clear the interrupt output before re-executing the WFI.

Alternatively, the software could disable the RAS fault and error interrupt outputs before executing the powerdown WFI so that the WFI is the point of no return for powering down the core. However, this would mean that any detected faults or errors encountered during the powerdown sequence would not be reported and the records of the fault or error would be lost.

4.7 Debug over powerdown

The C1-Ultra core supports debug over powerdown, which allows a debugger to retain its connection with the core even when powered down. This behavior enables debug to continue through powerdown scenarios rather than having to re-establish a connection each time the core is powered up.

The debug over powerdown logic is part of the DebugBlock in the C1-DynamIQ™ Shared Unit (DSU). The DebugBlock is external to the C1-DSU cluster and must remain powered on during the debug over powerdown process.

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

5. Memory management

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

This translation is based on address mapping and memory attribute information that is available in the C1-Ultra core internal registers and translation tables. The MMU also controls memory access permissions, memory ordering, and cache policies for each region of memory.

An address translation from an input address to an output address is described as a stage of address translation. The C1-Ultra core can perform:

- Stage 1 translations that translate an input Virtual Address (VA) to an output Physical Address (PA) or Intermediate Physical Address (IPA).
- Stage 2 translations that translate an input IPA to an output PA.
- Combined stage 1 and stage 2 translations that translate an input VA to an IPA, and then translate that IPA to an output PA. The C1-Ultra core performs translation table walks for each stage of the translation.

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

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

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

5.1 Memory Management Unit components

The C1-Ultra Memory Management Unit (MMU) includes several Translation Lookaside Buffers (TLBs), an MMU Translation Cache (MMUTC), and a translation table prefetcher.

A TLB is a cache of recently executed page translations within the MMU. The C1-Ultra core implements a two-level TLB structure.

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

The following table describes the MMU components.

Table 5-1: MMU components

Component	Description
L1 instruction TLB	• Caches entries at the 4KB, 16KB, 64KB, or 2MB granularity of Virtual Address (VA) to Physical Address (PA) mapping only • Fully associative • 128 entries
L1 data TLB	• Caches entries at the 4KB, 16KB, 64KB, 2MB, or 512MB granularity of VA to PA mappings only • Fully associative • 96 entries
L1 Statistical Profiling Extension (SPE) TLB	• Located in the SPE block • VA to PA translations of any page and block size • 1 entry
L1 TRace Buffer Extension (TRBE) TLB	• VA to PA translations of any page and block size • 1 entry
L2 TLB	• Shared by instructions and data • VA to PA mappings for 4KB, 16KB, 64KB, 2MB, 32MB, 512MB, and 1GB block sizes • Intermediate Physical Address (IPA) to PA mappings for: ◦ 2MB and 1GB block sizes in a 4KB translation granule ◦ 32MB block size in a 16KB translation granule ◦ 512MB block size in a 64KB granule • Intermediate PAs (descriptor PAs) obtained during a translation table walk • 8-way set associative • 2048 entries
Translation table prefetcher	• Detects access to contiguous translation tables and prefetches the next one • Can be disabled in the ECTLR register

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

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

L1 TLB Functionality

A hit in the L1 instruction TLB returns the PA to the instruction cache for comparison. It also checks the access permissions to signal an Instruction Abort.

A hit in the L1 data TLB returns the PA to the data cache for comparison. It also checks the access permissions to signal a Data Abort.

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

5.2 Translation Lookaside Buffer entry content

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

Each TLB entry contains:

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

Each TLB entry is associated with either:

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

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

Related information

5.4 Translation table walks on page 59

5.3 Translation Lookaside Buffer match process

The Arm®v9.3-A architecture supports multiple Virtual Address (VA) spaces that are translated differently.

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

- Secure EL3
- Secure EL2
- Secure EL2 and EL0
- Non-secure EL2
- Non-secure EL2 and EL0
- Secure EL1 and EL0
- Non-secure EL1 and EL0

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

- Its VA[63:N], where N is $\log_2$ of the block size for the translation that is stored in the TLB entry, matches the requested address. VA[63:56] is used for the comparison only if address tagging is not enabled.
- Entry translation regime matches the current translation regime.

- The Address Space IDentifier (ASID) matches the current ASID held in the TTBR0_ELx or TTBR1_ELx register associated with the target translation regime, or the entry is marked global.
- The Virtual Machine IDentifier (VMID) matches the current VMID held in the VTTBR_EL2 register.

The ASID information is used for the purpose of TLB matching for entries using:

- The Secure EL1 and EL0 and Non-secure EL1 and EL0 translation regime
- The Secure EL2 and EL0 and Non-secure EL2 and EL0 translation regime

The VMID information is used for the purpose of TLB matching for entries using:

- The Secure EL1 and EL0 and Non-secure EL1 and EL0 translation regime, when EL2 is enabled.

5.4 Translation table walks

When the C1-Ultra core generates a memory access, the Memory Management Unit (MMU) searches for the requested Virtual Address (VA) in the Translation Lookaside Buffers (TLBs). If it is not present, then it is a miss and the MMU proceeds by looking up the translation table during a translation table walk.

When the C1-Ultra core generates a memory access, the MMU:

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

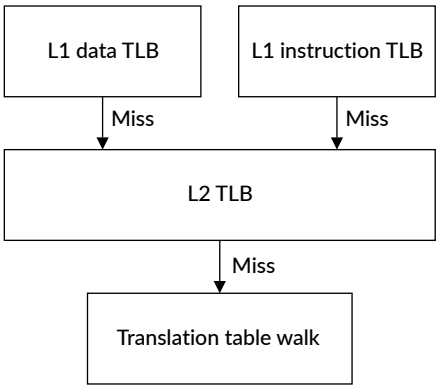
Address translation is performed only when the MMU is enabled. It can also be disabled for a particular translation base register, in which case the MMU returns a Translation Fault.

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

During a lookup or translation table walk, the access permission bits in the matching translation table entry determine whether the access is permitted. If the permission checks are violated, then the MMU returns a Permission Fault. For more information, see the *Arm® Architecture Reference Manual for A-profile architecture*.

The following figure shows the TLB lookup process.

Figure 5-1: Translation table walks



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

Related information

6. L1 instruction memory system on page 65
7. L1 data memory system on page 69
8. L2 memory system on page 74

5.5 Hardware management of the Access flag and dirty state

The C1-Ultra core includes the option to perform hardware updates to the translation tables.

This feature is enabled in TCR_ELx (where x is 1-3) and VTCR_EL2. To support hardware management of dirty state, translation table descriptors include the Dirty Bit Modifier (DBM) field.

The C1-Ultra core supports hardware updates to the Access flag and to dirty state only when the translation tables are held in Inner Write-Back and Outer Write-Back Normal memory regions. If software requests a hardware update in a region that is not Inner Write-Back or Outer Write-Back Normal memory, then the C1-Ultra core returns an abort with the following encoding:

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

5.6 Responses

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

MMU responses

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

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

The responses from the MMU contain the following information:

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

MMU aborts

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

External aborts

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

External aborts are reported synchronously when they occur during:

- Translation table walks for instruction fetches, loads, and stores
- Load operations to Inner Write-Back, Outer Write-Back Normal Cacheable memory

External aborts are reported asynchronously when they occur during:

- Load operations to all memory locations other than Inner Write-Back, Outer Write-Back Normal memory when the access is not caused by a translation table walk
- Store operations to any memory type
- Cache maintenance, TLB invalidate, and instruction cache invalidate operations
- Atomic operations including `AtomicLd`, `AtomicSt`, `AtomicCAS`, and `AtomicSwap`

C1-Ultra takes a synchronous abort on a Normal memory ldrx that receives an non-EXOK response from CHI. The abort is asynchronous for Device memory ldrx. For strx, OK and EXOK responses are expected and do not cause aborts. NDErr and DErr responses for WriteNoSnp Excl=1 cause asynchronous aborts.

Misprogramming contiguous hints

When there is a descriptor that contains a set CH bit, the input Virtual Address (VA) address space must include all contiguous VAs contained in this block. The C1-Ultra core treats the block as not causing a translation fault and disregards the value of the contiguous bit.

The VA address space is defined by:

- TCR_ELx.TxSZ for stage 1 translations
- VTCR_EL2.T0SZ for stage 2 translations

Conflict aborts

The C1-Ultra core does not generate conflict abort exceptions unless configured to do so via IMP_CPUECTLR_EL1.[26] (DTLB_CABT_EN). Generating conflict abort exceptions this way only applies to L1 Data TLB.

Arm strongly recommends that IMP_CPUECTLR_EL1.[26] (DTLB_CABT_EN) is not altered from its reset value, which disables conflict aborts.

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

5.7 Memory behavior and supported memory types

The C1-Ultra core supports memory types defined in the Armv8-A architecture.

Device memory types have the following attributes:

G – Gathering
: The capability to gather and merge requests together into a single transaction

R – Reordering
: The capability to reorder transactions

E – Early Write Acknowledgment
: The capability to accept early acknowledgment of write transactions from the interconnect


Note

In the following table, the n prefix means that the capability is not allowed for the memory type. For example, nGnRE means non-Gathering, non-Reordering, Early Write Acknowledgment.

The following table shows how memory types are supported in the C1-Ultra core.

Table 5-2: Supported memory types

Memory type	Shareability	Inner Cacheability	Outer Cacheability	Notes
Device nGnRnE	Any	-	-	-
Device nGnRE	Any	-	-	-
Device nGRE	Any	-	-	-
Device GRE	Any	-	-	-

Memory type	Shareability	Inner Cacheability	Outer Cacheability	Notes
Normal	Any	Non-cacheable	Any	-
Normal	Any	Write-Through Cacheable	Any	Treated as Non-cacheable
Normal	Any	Write-Back Cacheable	Non-cacheable	Treated as Non-cacheable
Normal	Any	Write-Back Cacheable	Write-Through Cacheable	Treated as Non-cacheable
Normal	Non-shareable	Write-Back Cacheable	Write-Back Cacheable	Treated as Non-cacheable
Normal	Inner/Outer Shareable	Write-Back Cacheable	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.
Normal	Inner/Outer Shareable	Write-Back Cacheable	Write-Back Cacheable (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.

Some behaviors are simplified. For best performance, Arm does not recommend using the following memory types:

Write-Through

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

Mixed Inner and Outer Cacheability

Only memory that is marked as Inner and Outer Write-Back can be cached on the data side and make coherency requests. This rule applies to the memory type only, and not to the allocation hints. All caches within the C1-DSU cluster are treated as being part of the Inner Cacheability domain.

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

5.8 Page-based hardware attributes

The architecture defines Page-Based Hardware Attributes (PBHA) as an optional **IMPLEMENTATION DEFINED** feature. This section describes how the C1-Ultra core implements PBHA.

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

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

For memory accesses caused by a translation table walk, the IMP_ATCR_ELx and IMP_AVTCR_EL2 registers control the PBHA values.

PBHA combination between stage 1 and stage 2 on memory accesses

PBHA should always be considered as an attribute of the physical address. Enable of PBHA has a granularity of 1 bit, so this property is applied independently on each PBHA bit.

When stage 1 and stage 2 are enabled:

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

Mismatched aliases

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

6. L1 instruction memory system

The C1-Ultra core L1 instruction memory system fetches instructions and predicts branches. It includes the L1 instruction cache, the L1 instruction Translation Lookaside Buffer (TLB), and the branch prediction unit.

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

The following table shows the L1 instruction memory system features.

Table 6-1: L1 instruction memory system features

Feature	Description
L1 instruction cache	• 64KB • 4-way set associative • Virtually Indexed, Physically Tagged (VIPT) behaving as Physically Indexed, Physically Tagged (PIPT) • Always protected with parity
Cache line length	64 bytes
Cache policy	L1 I-cache. Pseudo-Least Recently Used (LRU) cache replacement policy for L1
Interface with L2 memory system	32 bytes per cycle interface


Note

The L1 instruction TLB also resides in the L1 instruction memory system. However, it is part of the Memory Management Unit (MMU) and is described in 5. Memory management on page 56.

6.1 L1 instruction cache behavior

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

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

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

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

Note

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

Related information

4.4.5 Debug recovery mode on page 50

6.2 L1 instruction cache Speculative memory accesses

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

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

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

If the L1 instruction cache is enabled and the instruction fetches miss in the L1 instruction cache, then they will look up in L2 and snoop the L1 data cache if those caches are enabled. Instruction fetches are always coherent with data caches, so cache maintenance operations are not required to make stores visible to instruction fetches. However, the lookup never causes an L1 data cache refill, regardless of the data cache enable status. The line is only allocated in the L2 cache, provided that the L1 instruction cache is enabled.

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

6.3 Program flow prediction

The C1-Ultra core contains program flow prediction hardware, also known as branch prediction. Branch prediction increases overall performance and enhances power efficiency.

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

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

Program flow prediction hardware contains the following functionality:

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

Predicted and non-predicted instructions

Program flow prediction hardware predicts all branch instructions, and includes:

- Conditional branches
- Unconditional branches
- Return instructions
- Indirect branches

The following instructions are not predicted:

- Exception return instructions (including `ERET`, `ERETAA`, `ERETAB`)
- Supervisor call instructions
- Hypervisor call instructions
- Secure Monitor call instructions

Return stack

The return stack stores the return address of procedure call instructions. This address should be equal to the value written in the Link Register (X30) by these instructions.

Any of the following instructions causes a return stack push:

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

Any of the following instructions cause a return stack pop:

- `RET`
- `RETAA`
- `RETAB`

6.4 Instruction Prefetch

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

Preload instructions

The C1-Ultra core supports the AArch64 prefetch memory instructions, `PRFM PLI`, into the L1 instruction cache or L2 cache.

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

The `PRFM PLD` and `PRFM PST` instructions perform preloading in the L1 data cache, L2 cache, or L3 cache. `PRFM PLD` and `PRFM PST` instructions translate through the Data TLB.

The `PRFM PLI` instruction performs preloading to the L1 instruction cache and L2 cache. Instruction preloading is performed in the background. `PRFM PLI` instructions translate through the Instruction TLB.

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

7. L1 data memory system

The C1-Ultra core L1 data memory system executes load and store instructions. It services memory coherency requests and specific instructions such as atomics, cache maintenance operations, and memory tagging instructions. The L1 data memory system includes the L1 data cache and the L1 data Translation Lookaside Buffer (TLB).

The following table shows the L1 data memory system features.

Table 7-1: L1 data memory system features

Feature	Description
L1 data cache	128KB
	4-way set associative
	Virtually Indexed, Physically Tagged (VIPT) behaving as Physically Indexed, Physically Tagged (PIPT)
	Always protected with Error Correcting Code (ECC)
Cache line length	64 bytes
Cache policy	Re-Reference Interval Prediction (RRIP) replacement policy
Interface with integer execute pipeline and vector execute	• 4×64-bit read paths and 4×64-bit write paths for the integer execute pipeline • 4×128-bit read paths and 4×128-bit write paths for the vector execute pipeline


Note

The L1 data TLB also resides in the L1 data memory system. However, it is part of the Memory Management Unit (MMU) and is described in 5. Memory management on page 56.

7.1 L1 data cache behavior

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

In Debug recovery mode, the caches are not guaranteed to be functional and should not be enabled.

There is no operation to invalidate the entire data cache. If software requires this function, then it must be constructed by iterating over the cache geometry and executing a series of individual invalidates by set/way instructions.

The `DC CSW` and `DC ISW` instructions perform both a clean and invalidate of the target set/way. The values of HCR_EL2.SWIO have no effect. For more information about `DC CISW` and HCR_EL2, see the *Arm® Architecture Reference Manual for A-profile architecture*.

Data Cacheability disabled behavior

If the data Cacheability is disabled, then:

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

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

To maintain data coherency between multiple cores, the C1-Ultra core uses the Modified Exclusive Shared Invalid (MESI) protocol.


Note

The way that cache indices are determined means that there is no direct relationship between the Physical Address (PA) and set number. You cannot use targeted operations that assume a relationship between the PA and set number. To flush the entire cache, you must perform set and way maintenance operations over the number of sets and ways described in CCSIDR_EL1 for that cache.

Related information

4.4.5 Debug recovery mode on page 50

7.2 Write streaming mode

The C1-Ultra core supports write streaming mode, sometimes referred to as read allocate mode, both for the L1 and the L2 cache.

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

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

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


Note

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

The memory system continues in write streaming mode until either:

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

When a C1-Ultra core has switched to write streaming mode, the memory system continues to monitor the write traffic. It signals to the L1, L2, L3, and L4 cache to go into write streaming mode when it observes a further number of full cache line writes.

The write streaming threshold defines the number of consecutive cache lines that are fully written without being read before store operations stop causing cache allocations. You can configure the write streaming threshold for each cache (L1, L2, L3, and L4) by writing the register A.4.10 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 310.

7.3 Atomic instruction implementation in the L1 data memory system

The C1-Ultra core supports the atomic instructions added in the Arm®v8.1-A architecture.

Atomic instructions to Cacheable memory can be performed as either near atomics or far atomics, depending on where the cache line containing the data resides.

If an instruction hits in the L1 data cache, then the C1-Ultra core tries to perform it as a near atomic. Then, based on system behavior, the core can decide to perform it as a far atomic.

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

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

When Memory Tagging Extension (MTE) is enabled with precise checking, all checked atomics are performed near.

The C1-Ultra core supports atomics to Device or Non-cacheable memory. However, this relies on the interconnect also supporting atomics. If such an atomic instruction is executed when the interconnect does not support them, then it results in an abort.

7.4 Memory operations in the L1 data memory system

FEAT_MOPS is an Arm®v8.8-A feature that provides memory operation standardization.

The C1-Ultra core supports instructions that are optimized for the `memcpy()`, `memset()`, and `memmove()` family of functions. These instructions enable a standard optimized implementation across different microarchitectures. For more information about these instructions, see the *Arm® Architecture Reference Manual for A-profile architecture*.

The following table shows the instructions that are optimized for `memcpy()`, `memset()`, and `memmove()`.

Table 7-2: Optimized instructions for memcpy(), memset(), and memmove()

`memcpy()`, `memmove()`	`memset()`

`CPYFP [<Xd]!, [<Xs]!, <Xn!`
`CPYFM [<Xd]!, [<Xs]!, <Xn!`
`CPYFE [<Xd]!, [<Xs]!, <Xn!`
	`SETGP [<Xd>]!, <Xn>!, <Xs>`
	`SETGM [<Xd>]!, <Xn>!, <Xs>`
	`SETGE [<Xd>]!, <Xn>!, <Xs>`

• Three instruction sequence:
  ◦ `CPYP` performs pre-conditioning
  ◦ `CPYM` performs the operation
  ◦ `CPYE` finalizes the operation
• Exceptions can be taken part way through copy
• Options to control direction, whether accesses are temporal, and whether accesses are privileged
	• Three instruction sequence:
	  ◦ `SETP` performs pre-conditioning
	  ◦ `SETM` performs the operation
	  ◦ `SETE` finalizes the operation
	• Exceptions can be taken part way through set
	• Tag setting variants available
	• Options to control whether accesses are temporal, and whether accesses are privileged

Related information

7.2 Write streaming mode on page 70

7.5 Internal exclusive monitor

The C1-Ultra core includes an internal exclusive monitor with a 2-state, open and exclusive state machine that manages Load-Exclusive and Store-Exclusive accesses and Clear-Exclusive (`CLREX`) instructions.

You can use these instructions to construct semaphores, ensuring synchronization between different processes running on the core, and also between different cores that are using the same coherent memory locations for the semaphore. A Load-Exclusive instruction tags a small block

of memory for exclusive access. CTR_EL0 defines the size of the tagged blocks as 16 words, one cache line.


Note

A Load-Exclusive or Store-Exclusive instruction in the A64 instruction set is an instruction that has a mnemonic starting with LDX, LDAX, STX, or STLX.

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

For more technical reference and register information, see A.5.24 CTR_EL0, Cache Type Register on page 455.

7.6 Data prefetching

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

Hardware data prefetcher

The C1-Ultra core includes multiple hardware prefetch engines as part of the L1 and L2 caches. These prefetch engines prefetch data into the L1 and L2 caches using a combination of Virtual Address (VA) and Physical Address (PA).

The CPUECTLR registers allow control over some aspects of the prefetcher behavior. For more information, see:

- A.4.10 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 310
- A.4.11 IMP_CPUECTLR2_EL1, CPU Extended Control Register 2 on page 320

Data cache zero

In the C1-Ultra core, the Data Cache Zero by Virtual Address (DC ZVA) instruction sets a 64-byte block of memory, which is aligned to 64 bytes, to zero.

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

8. L2 memory system

The C1-Ultra core L2 memory system connects the core with the C1-DynamIQ™ Shared Unit (DSU) through the CPU bridge. It includes the private L2 cache.

The L2 cache is unified and private to each C1-Ultra core in a cluster.

The following table shows the L2 memory system features.

Table 8-1: L2 memory system features

Feature	Description
L2 cache	• 2MB 8-way set associative with 4 banks or 3MB 12-way set associative with 4 banks • Physically Indexed, Physically Tagged (PIPT) • Always protected with Error Correcting Code (ECC)
Cache line length	64 bytes
Cache policy	Dynamic biased cache replacement policy
Interface with the C1-DynamIQ™ Shared Unit (DSU)	One CHI Issue E compliant interfaces with 256-bit read and write DAT channel widths

8.1 L2 cache

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

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

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


Note

The way that cache indices are determined means that there is no direct relationship between the Physical Address (PA) and set number. You cannot use targeted operations that assume a relationship between the PA and set number. To flush the entire cache, you must perform set and way maintenance operations over the number of sets and ways described in CCSIDR_EL1 for that cache. This operation is compliant with the Armv8-A architecture.

Related information

4.4.5 Debug recovery mode on page 50

8.2 Support for memory types

The C1-Ultra core simplifies coherency logic by downgrading some memory types.

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

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

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

The additional attribute hints are used as follows:

Allocation hint

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

Transient hint

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

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

8.3 Transaction capabilities

The interface between the C1-Ultra core L2 memory system and the C1-DynamIQ™ Shared Unit (DSU) provides transaction capabilities for the core.

The following table shows the maximum possible values for read, write, Distributed Virtual Memory (DVM) issuing, and snoop capabilities of the C1-Ultra core L2 cache.

Table 8-2: C1-Ultra core transaction capabilities

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

For information on the different memory types, see the *Arm® Architecture Reference Manual for A-profile architecture*.

9. Direct access to internal memory

The C1-Ultra core provides a mechanism to read the internal memory that the L1 caches, L2 cache, and Translation Lookaside Buffer (TLB) structures use through **IMPLEMENTATION DEFINED** System registers. When the coherency between the cache data and the system memory data is broken, you can use this mechanism to investigate any issues.


Note

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

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

You can access the contents of the internal memory using the six Read-Only (RO) System registers in Table 9-1: System registers used to access internal memory on page 76. The internal memory is selected by programing the **IMPLEMENTATION DEFINED** RAMINDEX system instruction. Data Synchronization Barrier (DSB) and Instruction Synchronization Barrier (ISB) instructions are required between a write to the RAMINDEX register and the subsequent read of a DDATA or IDATA register.

Program the **IMPLEMENTATION DEFINED** RAMINDEX system instruction using the following `SYS` instruction:

```
SYS #6, C15, C0, #0, <Xt>
DSB SY
ISB
```

For more information on the RAMINDEX register, see A.13.2 SYS_IMP_RAMINDEX, RAM Index on page 588. The data is read from the read-only System registers as shown in the following table.


Note

- All the System registers are RO and 64-bits wide
- For the register reset value, see the individual bit resets
- Any access to the data registers returns data
- Click the register name for details on the returned data format

Table 9-1: System registers used to access internal memory

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

Register name	Function	Access	Operation	Rd Data
IMP_DDATA1_EL3	Data register 1	RO	`MRS <Xd>, S3_6_c15_c1_1`	Data
IMP_DDATA2_EL3	Data register 2	RO	`MRS <Xd>, S3_6_c15_c1_2`	Data

9.1 L1 cache encodings

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

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

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

Table 9-2: C1-Ultra L1 instruction cache tag location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x00`
[23:20]	Reserved
[19:18]	Way
[17:14]	Reserved
[13:6]	Virtual address [13:6]
[5:0]	Reserved

Table 9-3: C1-Ultra L1 instruction cache data location encoding

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

Table 9-4: C1-Ultra L1 instruction TLB data location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x04`
[23:8]	Reserved
[7:0]	TLB entry (0-127)

Table 9-5: C1-Ultra L1 data cache tag location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x08`
[23:20]	Reserved

Bit field of Xn	Description
[19:18]	Way
[17:16]	Copy: **0b00** Tag RAM associated with Pipe 0 **0b01** Tag RAM associated with Pipe 1 **0b10** Reserved **0b11** Reserved
[15]	Reserved
[14:6]	Virtual address [14:6]
[5:0]	Reserved

Table 9-6: C1-Ultra L1 data cache data location encoding

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

Table 9-7: C1-Ultra L1 data TLB location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x0A`
[23:7]	Reserved
[6:0]	TLB Entry (0-95)

9.1.1 L1 instruction tag RAM returned data

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

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

Table 9-8: L1 instruction cache tag format for Instruction Register 0

Bit field	Description
[63:32]	Reserved
[31]	Non-secure identifier for the physical address
[30:3]	Physical address [39:12]

Bit field	Description
[2:1]	Instruction state [1:0] **0b00** Invalid **0b01** Valid **0b10** Hardware prefetch (HWPRF) **0b11** Valid
[0]	Parity

Table 9-9: L1 instruction cache tag format for Instruction Register 1

Bit field	Description
[63:0]	0

Table 9-10: L1 instruction cache tag format for Instruction Register 2

Bit field	Description
[63:0]	0

9.1.2 L1 instruction data RAM returned data

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

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

Table 9-11: L1 instruction cache data format for Instruction Register 0

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

Table 9-12: L1 instruction cache data format for Instruction Register 1

Bit field	Description
[63:12]	0
[11:0]	Data [75:64]

Table 9-13: L1 instruction cache data format for Instruction Register 2

Bit field	Description
[63:0]	0

9.1.3 L1 instruction TLB returned data

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

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

Table 9-14: L1 instruction TLB format for Instruction Register 0

Bit field	Description
[63]	Virtual address [12]
[62:59]	PBHA [3:0]
[58]	Reserved
[57:55]	Memory attributes: **0b000** Device nGnRnE **0b001** Device nGnRE **0b010** Device nGRE **0b011** Device GRE **0b100** Non-cacheable **0b101** Write-Back No-Allocate **0b110** Write-Back Transient **0b111** Write-Back Read-Allocate and Write-Allocate
[54:52]	Page size: **0b000** 4KB **0b001** 16KB **0b010** 64KB **0b100** 2MB **Other** Reserved
[51]	Outer-shared
[50]	Inner-shared
[49:42]	Reserved
[41:40]	Reserved

Bit field	Description
[39:24]	ASID[15:0]
[23:8]	VMID[15:0]
[7:5]	Translation Regime[2:0]: **0b000** Secure EL1/EL0 **0b001** Secure EL2 **0b101** Secure EL3 **0b010** Non-secure EL1/EL0 **0b011** Non-secure EL2
[4:1]	Reserved
[0]	Valid

Table 9-15: L1 instruction TLB format for Instruction Register 1

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

Table 9-16: L1 instruction TLB format for Instruction Register 2

Bit field	Description
[63:4]	Reserved
[3]	Reserved
[2]	Reserved
[1]	Reserved
[0]	Non-Secure

9.1.4 L1 data tag RAM returned data

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

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

Table 9-17: L1 data cache tag format for Data Register 0

Bit field	Description
[63:57]	*Error Correcting Code* (ECC)
[56:28]	Non-secure identifier, physical address [39:12]
[27:25]	Reserved
[24]	Transient/WBNA

Bit field	Description
[23:20]	Memory Tagging Extension (MTE) tag poison
[19:4]	MTE tag data
[3:2]	MTE tag state: **0b00** Invalid **0b01** Shared **0b11** Dirty state
[1:0]	MESI: **0b00** Invalid **0b01** Shared **0b10** Exclusive **0b11** Modified

Table 9-18: L1 data cache tag format for Data Register 1

Bit field	Description
[63:0]	0

Table 9-19: L1 data cache tag format for Data Register 2

Bit field	Description
[63:0]	0

9.1.5 L1 data cache data RAM returned data

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

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

Table 9-20: L1 data cache data format for Data Register 0

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

Table 9-21: L1 data cache data format for Data Register 1

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

Table 9-22: L1 data cache data format for Data Register 2

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

9.1.6 L1 data TLB returned data

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

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

Table 9-23: L1 data TLB format for Data Register 0

Bit field	Description
[63:62]	LOR ID [1:0]
[61]	LOR match
[60]	Outer-shared
[59]	Inner-shared
[58:57]	S1 translation regime [1:0]
[56:55]	S2 translation regime [1:0]
[54:52]	Memory attributes [2:0]: **0b000** Device nGnRnE **0b001** Device nGnRE **0b010** Device nGRE **0b011** Device GRE **0b100** Non-cacheable **0b101** Write-Back No-Allocate **0b110** Write-Back Transient **0b111** Write-Back Read-Allocate and Write-Allocate
[51]	Outer allocate
[50]	S2 Dirty Bit Modifier (DBM) bit
[49]	S1 DBM bit
[48]	TLB coalesced bit
[47:44]	Permission bit [3:0]

Bit field	Description
[43]	Device/Non-cacheable HTRAP
[42]	nG bit
[41]	Smash bit - Indicates the page or block has been fractured.
[40:38]	Page size [2:0]: **0b000** 4KB **0b001** 16KB **0b010** 64KB **0b011** Reserved **0b100** 2MB **0b101** Reserved **0b110** 512MB **0b111** Reserved
[37:36]	Security State **00** Secure **01** Non-secure **10** Root **11** Realm
[35:33]	MSID [1:0]
[32:17]	ASID [15:0]
[16:1]	VMID [15:0]
[0]	Valid

Table 9-24: L1 data TLB format for Data Register 1

Bit field	Description
[63:37]	Physical address [38:12]
[36:0]	Virtual address [48:12]

Table 9-25: L1 data TLB format for Data Register 2

Bit field	Description
[8]	Tagged MTE

Bit field	Description
[7:6]	Reserved
[5]	FWB override
[4]	PBHA [3]
[3]	PBHA [2]
[2]	PBHA [1]
[1]	PBHA [0]
[0]	PA [39]

9.2 L2 cache encodings

The 2MB L2 cache is 8-way set associative and the 3MB L2 cache is 12-way set associative.

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

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


Note

The following applies for all tables:

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

Table 9-26: C1-Ultra L2 cache tag location encoding for 2MB

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

Table 9-27: C1-Ultra L2 cache tag location encoding for 3MB

Bit field of Xn	Description
[31:24]	RAMID = `0x10`
[23:22]	Reserved
[21:18]	Way (0-11)
[17:12]	Index[17:12]

Bit field of Xn	Description
[11:8]	XOR(Index[11:8], Way[3:0])
[7:6]	XOR(Index[7:6], Index[11:10], Way[3:2])
[5:0]	Reserved

Table 9-28: C1-Ultra L2 cache data location encoding for 2MB

Bit field of Xn	Description
[31:24]	RAMID = `0x11`
[23:21]	Reserved
[20:18]	Way(0-7)
[17:11]	XOR(Index[17:11], 7'b0001000)
[10:8]	XOR(Index[10:8], Way[2:0])
[7]	XOR(Index[7], Index[11])
[6]	XOR(Index[6], Index[10], Way[2])
[5:4]	Physical address[5:4]
[3:0]	Reserved

Table 9-29: C1-Ultra L2 cache data location encoding for 3MB

Bit field of Xn	Description
[31:24]	RAMID = `0x11`
[23:22]	Reserved
[21:18]	Way (0-11)
[17:12]	XOR(Index[17:12], 6'b000100)
[11:8]	XOR(Index[11:8], Way[3:0])
[7:6]	XOR(Index[7:6], Index[11:10], Way[3:2])
[5:4]	Physical address[5:4]
[3:0]	Reserved

Table 9-30: C1-Ultra L2 TLB location encoding

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

Table 9-31: C1-Ultra 2MB L2 victim location encoding

Bit field of Rd	Description
[31:24]	RAMID = `0x12`
[23:18]	Reserved
[17:8]	XOR(Index[17:8], 10'b0010000000)
[7:6]	XOR(Index[7:6], Index[11:10])

Bit field of Rd	Description
[5:0]	Reserved

Table 9-32: C1-Ultra 3MB L2 victim location encoding

Bit field of Rd	Description
[31:24]	RAMID = `0x12`
[23:18]	Reserved
[17:8]	XOR(Index[17:8], 10'b0010000000)
[7:6]	XOR(Index[7:6], Index[11:10])
[5:0]	Reserved

9.2.1 L2 tag RAM returned data

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

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

For 2MB or 3MB L2 cache

n=33, m=18

Table 9-33: L2 tag cache format for Data Register 0

Bit field	Description
[63:n+20]	0
[n+19:n+13]	ECC
[n+12]	MPAM_PMG
[n+11:n+6]	MPAM_PARTID
[n+5]	MPAM_NS
[n+4:n+1]	PBHA[3:0]
[n:12]	Physical tag [39:m]
[11]	Non-secure
[10]	CopyAtHome
[9:7]	Virtual address [14:12]
[6]	Shareable
[5]	L1 data cache valid
[4:3]	MTE state: **`0b00`** Invalid **`0b01`** Clean **`0b11`** Dirty

Bit field	Description
[2:0]	L2 state: **0b101** UniqueDirty **0b001** UniqueClean **0bx11** SharedClean **0bxx0** Invalid

Table 9-34: L2 tag cache format for Data Register 1

Bit field	Description
[63:0]	0

Table 9-35: L2 tag cache format for Data Register 2

Bit field	Description
[63:0]	0

9.2.2 L2 data RAM returned data

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

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

Table 9-36: L2 data RAM format for Data Register 0

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

Table 9-37: L2 data RAM format for Data Register 1

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

Table 9-38: L2 data RAM format for Data Register 2 when L2 ECC granule equals 128

Bit field	Description
[63:15]	0
[14:6]	ECC[8:0] for Data [127:0]
[5]	Poison for Data [127:64]
[4]	Poison for Data [63:0]
[3:0]	MTE tags

Table 9-39: L2 data RAM format for Data Register 2 when L2 ECC granule equals 256

Bit field	Description
[63:11]	0
[10:6]	ECC[4:0] for Data [255:0] **Note:** ECC[9:5] for Data[255:0] are contained in the read of the next quad-word.
[5]	Poison for Data [127:64]
[4]	Poison for Data [63:0]
[3:0]	MTE tags

9.2.3 L2 TLB RAM returned data

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

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

L2 TLB format for Instruction Register 0 will have different physical addresses for bits [20:53] if bit[6] is set to 1 or 0.

Table 9-40: L2 TLB format for Instruction Register 0 if bit[6] is set to 0

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

Bit field	Description
[53:20]	Physical address When bit[6] is 0: • [53:26] = PA[39:12] • [25:20] = Reserved
[19:17]	Page size: **0b000** 4KB **0b001** 16KB **0b010** 64KB **0b100** 2MB **0b101** 32MB **0b110** 512MB **0b111** 1GB
[16:7]	Reserved
[6]	Coalesced entry
[5:2]	Valid bits
[1:0]	Reserved

Table 9-41: L2 TLB format for Instruction Register 0 if bit[6] is set to 1

Bit field	Description
[63:62]	Reserved
[61:20]	Physical address When bit[6] is 1: • [53:28] = PA[39:14] • [27:26] = PA[13:12] for page 3 (highest memory address) • [25:24] = PA[13:12] for page 2 • [23:22] = PA[13:12] for page 1 • [21:20] = PA[13:12] for page 0 (lowest memory address)

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

Table 9-42: L2 TLB format for Instruction Register 1

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

Table 9-43: L2 TLB format for Instruction Register 2

Bit field	Description
[63:22]	Reserved

Bit field	Description
[21:19]	Translation Regime [2:0]: **0b000** Secure EL1 **0b001** Secure EL2 **0b010** Non-secure EL1 **0b011** Non-secure EL2 **0b101** EL3
[18:3]	VMID [15:0]
[2:0]	ASID [15:13]

9.2.4 L2 Victim RAM returned data

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

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

Table 9-44: C1-Ultra L2 victim format for data register 0 for 2MB L2 cache

Bit field of Rd	Description
[63:56]	Prefetch
[55:48]	Data source
[47:40]	Transient
[39:32]	Outer allocation hint
[31:16]	Pointer fill counter
[15:0]	Replacement

Table 9-45: C1-Ultra L2 victim format for data register 0 for 3MB L2 cache

Bit field of Rd	Description
[63:60]	Transient[3:0]
[59:48]	Outer allocation hint
[47:24]	Pointer fill counter
[23:0]	Replacement

Table 9-46: C1-Ultra L2 victim format for data register 1 for 2MB L2 cache

Bit field of Rd	Description
[63:0]	0

Table 9-47: C1-Ultra L2 victim format for data register 1 for 3MB L2 cache

Bit field of Rd	Description
[63:32]	0
[31:20]	Prefetch
[19:8]	Source
[7:0]	Transient[11:4]

Table 9-48: C1-Ultra L2 victim format for data register 2

Bit field of Rd	Description
[63:0]	0

10. RAS extension support

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

In particular, the C1-Ultra core supports these RAS Extension features:

- Fault Handling Interrupts (FHIs)
- Error Recovery Interrupts (ERIs)
- Poison attribute on bus transfers
- Cache protection with Single Error Correct (SEC) parity on the functional RAMs that only contain clean data. This includes the L1 instruction cache tag, L1 instruction cache data, and the Memory Management Unit (MMU) RAMs.
- Cache protection with Single Error Correct, Double Error Detect (SECDED), Error Correcting Code (ECC) on the functional RAMs that contain dirty data. This includes the L1 data cache tag, L1 data cache data, L2 cache tag, L2 cache data, and the L2 Transaction Queue (TQ) RAMs.
- Error Data 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 core ensures that all SError interrupts that are generated by instructions before the `ESB` are either taken or deferred. If the core cannot take the interrupt, it records 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 and the C1-Ultra core. 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 following nodes are implemented in the C1-Ultra core and the C1-DSU cluster:

- Node 0 includes the shared L3 memory system in the C1-DynamIQ™ Shared Unit (DSU).
- Node 1 includes the private L1 memory system, L2 memory system, and the MMU/TLB in the core.
- When a single C1-SME2 unit is implemented in the cluster, Node 2 includes the C1-SME2 memory system, context, and prefetcher memories.
- When two C1-SME2 units are implemented in the cluster, Node 3 includes the memory system, context, and prefetcher memories for the second C1-SME2 instance.

For more information on the architectural RAS Extension and the definition of a node, see the *Arm® Architecture Reference Manual for A-profile architecture*.

For information on the node that includes the shared L3 memory system, see *RAS extension support* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

For more information on C1-SME2, see the *Arm® C1-Scalable Matrix Extension 2 Technical Reference Manual*.

10.1 Cache protection behavior

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

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

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

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

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

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

10.2 Error containment

The C1-Ultra core supports error containment for data errors. This means that detected data errors are not silently propagated. Data errors are deferred using data poisoning to ensure that you are aware of the error.

The following error types are not containable:

- Uncorrectable L1 data cache tag errors
- Uncorrectable L1 data cache dirty errors
- Uncorrectable L2 cache tag errors

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

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

10.3 Fault detection and reporting

When the C1-Ultra core detects a fault, it raises a Fault Handling Interrupt (FHI) exception or an Error Recovery Interrupt (ERI) exception through the fault or the error signals. FHIs and ERIs are reflected in the Reliability, Availability, and Serviceability (RAS) registers, which are updated in the node that detects the errors.

Fault handling interrupts

When 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 core n are signaled using nCOREFAULTIRQ[n].

Error recovery interrupts

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

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

Related information

B.8 External RAS registers summary on page 834
B.8.2 ERR0CTLR, Error Record <n> Control Register on page 838
B.8.3 ERR0STATUS, Error Record <n> Primary Status Register on page 840

10.4 Error detection and reporting

When the C1-Ultra core consumes an error, it raises different exceptions depending on the error type.

The C1-Ultra core might raise:

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

10.4.1 Error reporting and performance monitoring

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

The Performance Monitoring Unit (PMU) event counters count the MEMORY_ERROR event if it is selected and the counter is enabled.

In Secure state, the MEMORY_ERROR event is counted only if MDCR_EL3.SPME is asserted. For a description of MDCR_EL3, see the *Arm® Architecture Reference Manual for A-profile architecture*.

Related information
17.1 Common PMU events on page 125

10.5 Error injection

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

Error injection uses the error detection and reporting registers to insert errors. The C1-Ultra core can inject the following error types:

Corrected errors
A Corrected Error (CE) is generated for a single-bit Error Correcting Code (ECC) error injection.

Deferred errors
A Deferred Error (DE) is generated for a double-bit ECC error injection.

Uncontainable errors
An Uncontainable Error (UC) is generated for a uncontainable error injection.

The C1-Ultra core supports the ERXPFGF_EL1.NA bit, meaning that error and fault injections are generated without performing a memory access. For more information about ERXPFGF_EL1.NA, see B.8.9 ERR0PFGF, Error Record <n> Pseudo-fault Generation Feature Register on page 864.

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.

10.6 AArch64 RAS registers

The following summary table provides an overview of all RAS registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 10-1: RAS registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ERRIDR_EL1	3	0	C5	C3	0	0x0000000000000002	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	0x005100008010A9A2	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	0x0000000040000062	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
DISR_EL1	3	0	C12	C1	1	See individual bit resets.	64-bit	Deferred Interrupt Status Register
VSESR_EL2	3	4	C5	C2	3	See individual bit resets.	64-bit	Virtual SError Exception Syndrome Register
VDISR_EL2	3	4	C12	C1	1	See individual bit resets.	64-bit	Virtual Deferred Interrupt Status Register (EL2)

10.7 External RAS registers

The following summary table provides an overview of all 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.

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 10-2: RAS registers summary

Offset	Name	Reset	Width	Description
0x0	ERR0FR	0x005100008010A9A2	64-bit	Error Record <n> Feature Register

Offset	Name	Reset	Width	Description
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	0x0000000040000062	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	0xD8C1083B	32-bit	Implementation Identification Register
0xFA8	ERRDEVAFF	See individual bit resets.	64-bit	Device Affinity Register
0xFBC	ERRDEVARCH	See individual bit resets.	32-bit	Device Architecture Register
0xFC8	ERRDEVID	0x00000001	32-bit	Device Configuration Register
0xFD0	ERRPIDR4	0x00000004	32-bit	Peripheral Identification Register 4
0xFE0	ERRPIDR0	0x0000008C	32-bit	Peripheral Identification Register 0
0xFE4	ERRPIDR1	0x000000BD	32-bit	Peripheral Identification Register 1
0xFE8	ERRPIDR2	See individual bit resets.	32-bit	Peripheral Identification Register 2
0xFEC	ERRPIDR3	See individual bit resets.	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

11. Utility bus

The utility bus provides access to control registers for various system components in the C1-DynamIQ™ Shared Unit (DSU), the C1-SME2 unit, and the cores 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-Ultra core:

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

---

Note

For additional information about the Power Policy Units (PPU) registers for the cores and the C1-SME2 in the cluster, see the *External Core PPU registers summary* section of *External registers* 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*.

11.1 Base addresses for system components

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

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

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

Note

- The base address for each set of registers for the core RAS, AMU, and MPMM registers depend on the core instance number <n>, from 0 to the total number of cores minus one.
- 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.

Table 11-1: Utility bus base addresses for system component registers

Base address, n is core instance number	Registers	Security state	Memory map
0x<n>9_0000	Core <n> AMU	Secure	B.1 External AMU registers summary on page 622
0x<n>A_0000	Core <n> RAS	Secure	B.8 External RAS registers summary on page 834
0x<n>B_0000	Core <n> MPMM	Secure	B.7 External PPM registers summary on page 821
0x<n>D_0000 - 0x<n>F_0000	Reserved	-	-

12. GIC CPU interface

The Generic Interrupt Controller (GIC) supports and controls interrupts. The GIC Distributor connects to the C1-Ultra core through a GIC CPU interface. The GIC CPU interface includes registers to mask, identify, and control the state of interrupts that are forwarded to the core.

Each core in a C1-DSU cluster has a GIC CPU interface, which connects to a common external Distributor component.

The GICv4.2 architecture implemented in the C1-Ultra core supports:

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

The GIC includes interrupt grouping functionality that supports:

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

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

12.1 Disable the GIC CPU interface

The C1-Ultra core always includes the Generic Interrupt Controller (GIC) CPU interface. However, you can disable it to meet your requirements.

To disable the GIC CPU interface, assert the GICCDISABLE signal HIGH at reset. If you disable it this way, then you can use an external GIC IP to drive the interrupt signals (nFIQ, nIRQ). If the C1-Ultra core is not integrated with an external GIC interrupt Distributor component (minimum GICv3 architecture) in the system, then you must disable the GIC CPU interface.

If you disable the GIC CPU interface, then:

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


Note

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

For more information on these signals, see *Functional integration* in the *Arm® C1-DynamIQ™ Shared Unit Configuration and Integration Manual*.

12.2 AArch64 GIC system registers

The following summary table provides an overview of all GIC system registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 12-1: GIC system registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICC_PMR_EL1	3	0	C4	C6	0	See individual bit resets.	64-bit	Interrupt Controller Interrupt Priority Mask Register
ICV_PMR_EL1	3	0	C4	C6	0	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Priority Mask Register
ICC_IAR0_EL1	3	0	C12	C8	0	See individual bit resets.	64-bit	Interrupt Controller Interrupt Acknowledge Register 0
ICV_IAR0_EL1	3	0	C12	C8	0	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 0
ICC_EOIR0_EL1	3	0	C12	C8	1	See individual bit resets.	64-bit	Interrupt Controller End Of Interrupt Register 0
ICV_EOIR0_EL1	3	0	C12	C8	1	See individual bit resets.	64-bit	Interrupt Controller Virtual End Of Interrupt Register 0
ICC_HPPIR0_EL1	3	0	C12	C8	2	See individual bit resets.	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 0
ICV_HPPIR0_EL1	3	0	C12	C8	2	See individual bit resets.	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
ICC_BPR0_EL1	3	0	C12	C8	3	See individual bit resets.	64-bit	Interrupt Controller Binary Point Register 0
ICV_BPR0_EL1	3	0	C12	C8	3	See individual bit resets.	64-bit	Interrupt Controller Virtual Binary Point Register 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICC_AP0R0_EL1	3	0	C12	C8	4	See individual bit resets.	64-bit	Interrupt Controller Active Priorities Group 0 Registers
ICV_AP0R0_EL1	3	0	C12	C8	4	See individual bit resets.	64-bit	Interrupt Controller Virtual Active Priorities Group 0 Registers
ICC_AP1R0_EL1	3	0	C12	C9	0	See individual bit resets.	64-bit	Interrupt Controller Active Priorities Group 1 Registers
ICV_AP1R0_EL1	3	0	C12	C9	0	See individual bit resets.	64-bit	Interrupt Controller Virtual Active Priorities Group 1 Registers
ICC_NMIAR1_EL1	3	0	C12	C9	5	See individual bit resets.	64-bit	Interrupt Controller Non-maskable Interrupt Acknowledge Register 1
ICV_NMIAR1_EL1	3	0	C12	C9	5	See individual bit resets.	64-bit	Interrupt Controller Virtual Non-maskable Interrupt Acknowledge Register 1
ICC_DIR_EL1	3	0	C12	C11	1	See individual bit resets.	64-bit	Interrupt Controller Deactivate Interrupt Register
ICV_DIR_EL1	3	0	C12	C11	1	See individual bit resets.	64-bit	Interrupt Controller Deactivate Virtual Interrupt Register
ICC_RPR_EL1	3	0	C12	C11	3	See individual bit resets.	64-bit	Interrupt Controller Running Priority Register
ICV_RPR_EL1	3	0	C12	C11	3	See individual bit resets.	64-bit	Interrupt Controller Virtual Running Priority Register
ICC_SGI1R_EL1	3	0	C12	C11	5	See individual bit resets.	64-bit	Interrupt Controller Software Generated Interrupt Group 1 Register
ICC_ASGI1R_EL1	3	0	C12	C11	6	See individual bit resets.	64-bit	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
ICC_SGI0R_EL1	3	0	C12	C11	7	See individual bit resets.	64-bit	Interrupt Controller Software Generated Interrupt Group 0 Register
ICC_IAR1_EL1	3	0	C12	C12	0	See individual bit resets.	64-bit	Interrupt Controller Interrupt Acknowledge Register 1
ICV_IAR1_EL1	3	0	C12	C12	0	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 1
ICC_EOIR1_EL1	3	0	C12	C12	1	See individual bit resets.	64-bit	Interrupt Controller End Of Interrupt Register 1
ICV_EOIR1_EL1	3	0	C12	C12	1	See individual bit resets.	64-bit	Interrupt Controller Virtual End Of Interrupt Register 1
ICC_HPPIR1_EL1	3	0	C12	C12	2	See individual bit resets.	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 1
ICV_HPPIR1_EL1	3	0	C12	C12	2	See individual bit resets.	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
ICC_BPR1_EL1	3	0	C12	C12	3	See individual bit resets.	64-bit	Interrupt Controller Binary Point Register 1
ICV_BPR1_EL1	3	0	C12	C12	3	See individual bit resets.	64-bit	Interrupt Controller Virtual Binary Point Register 1
ICC_CTLR_EL1	3	0	C12	C12	4	See individual bit resets.	64-bit	Interrupt Controller Control Register (EL1)
ICV_CTLR_EL1	3	0	C12	C12	4	See individual bit resets.	64-bit	Interrupt Controller Virtual Control Register
ICC_SRE_EL1	3	0	C12	C12	5	See individual bit resets.	64-bit	Interrupt Controller System Register Enable Register (EL1)
ICC_IGRPEN0_EL1	3	0	C12	C12	6	See individual bit resets.	64-bit	Interrupt Controller Interrupt Group 0 Enable Register
ICV_IGRPEN0_EL1	3	0	C12	C12	6	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Group 0 Enable Register
ICC_IGRPEN1_EL1	3	0	C12	C12	7	See individual bit resets.	64-bit	Interrupt Controller Interrupt Group 1 Enable Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICV_IGRPEN1_EL1	3	0	C12	C12	7	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Group 1 Enable Register
ICH_AP0R0_EL2	3	4	C12	C8	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Active Priorities Group 0 Registers
ICH_AP1R0_EL2	3	4	C12	C9	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Active Priorities Group 1 Registers
ICC_SRE_EL2	3	4	C12	C9	5	See individual bit resets.	64-bit	Interrupt Controller System Register Enable Register (EL2)
ICH_HCR_EL2	3	4	C12	C11	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Control Register
ICH_VTR_EL2	3	4	C12	C11	1	0x0000000090280003	64-bit	Interrupt Controller VGIC Type Register
ICH_MISR_EL2	3	4	C12	C11	2	0x0000000000000000	64-bit	Interrupt Controller Maintenance Interrupt State Register
ICH_EISR_EL2	3	4	C12	C11	3	See individual bit resets.	64-bit	Interrupt Controller End of Interrupt Status Register
ICH_ELRSR_EL2	3	4	C12	C11	5	See individual bit resets.	64-bit	Interrupt Controller Empty List Register Status Register
ICH_VMCR_EL2	3	4	C12	C11	7	See individual bit resets.	64-bit	Interrupt Controller Virtual Machine Control Register
ICH_LR0_EL2	3	4	C12	C12	0	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICH_LR1_EL2	3	4	C12	C12	1	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICH_LR2_EL2	3	4	C12	C12	2	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICH_LR3_EL2	3	4	C12	C12	3	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICC_CTLR_EL3	3	6	C12	C12	4	See individual bit resets.	64-bit	Interrupt Controller Control Register (EL3)
ICC_SRE_EL3	3	6	C12	C12	5	See individual bit resets.	64-bit	Interrupt Controller System Register Enable Register (EL3)
ICC_IGRPEN1_EL3	3	6	C12	C12	7	See individual bit resets.	64-bit	Interrupt Controller Interrupt Group 1 Enable Register (EL3)

13. Advanced SIMD and floating-point support

The C1-Ultra core supports the Advanced Single Instruction Multiple Data (SIMD) and scalar floating-point instructions in the A64 instruction set without floating-point exception trapping.

The C1-Ultra core floating-point implementation includes features up to Arm®v9.3-A. BFloat16 floating-point and Int8 matrix multiplication are part of these supported features.

The C1-Ultra core implements all operations in hardware with support for all combinations of:

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

The C1-Ultra core supports Alternate Floating Point (FEAT_AFP) behavior, as part of Arm®v8.7-A.

14. Scalable Vector Extensions support

The C1-Ultra core supports the Scalable Vector Extension (SVE), the Scalable Vector Extension 2 (SVE2), the Scalable Matrix Extension (SME), and the Scalable Matrix Extension 2 (SME2).

SVE and SVE2 are intended to complement, not replace, AArch64 Advanced SIMD and floating-point functionality.

SME and SME2 are extensions to SVE2, adding instructions and architectural state storage. These extensions introduce a Streaming SVE mode.

The C1-SME2 unit

The C1-SME2 unit is a shared unit that implements SME and SME2. The C1-SME2 unit is implemented inside a C1-DSU cluster. It is shared 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.

For more information on the C1-SME2 unit, see the *Arm® C1-Scalable Matrix Extension 2 Technical Reference Manual*.

14.1 SVE features

Scalable Vector Extension (SVE) is an extension introduced by the Armv8.2 architecture.

The key features that SVE provides are:

- Predication
- Gather-load and scatter-store
- Software-managed speculative vectorization

The C1-Ultra core implements a scalable vector length of 128 bits.

All the features and additions that SVE and SVE2 introduce are described in the *Arm® Architecture Reference Manual for A-profile architecture*.

14.2 Streaming SVE

The Scalable Matrix Extension (SME) and Scalable Matrix Extension 2 (SME2) define a specific vector length which can be different from the core Scalable Vector Extension (SVE) Vector Length (VL). A Streaming SVE (SSVE) mode is introduced for this purpose.

In normal mode, SVE instructions execute with the VL. The specific SME/SME2 instructions are **UNDEFINED** and the additional architectural state is not accessible.

In SSVE mode, SVE instructions execute with the Streaming SVE Vector Length (SVL) of 512-bits. When enabled in this mode, the specific SME/SME2 instructions and a subset of the Neon™ and SVE instructions can be executed, and the additional architectural state is accessible.

PSTATE.SM controls the Streaming SVE mode. PSTATE.ZA controls access to the SME ZA storage. These PSTATE fields can be modified by the SMSTART and SMSTOP instructions, and can also be read and written using the SVCR register.

The SMSTART instruction is used to enter Streaming SVE mode, or to enable the SME ZA storage, or both simultaneously.

The SMSTOP instruction is used to exit Streaming SVE mode, or to disable the SME ZA storage, or both simultaneously.

For more information, see the *Arm® Architecture Reference Manual for A-profile architecture* and the *Arm® C1-Scalable Matrix Extension 2 Technical Reference Manual*.

Related information

A.5.4 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0 on page 406
A.5.5 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1 on page 409
A.5.7 ID_AA64ZFR0_EL1, SVE Feature ID Register 0 on page 414
B.4.5 EDPFR, External Debug Processor Feature Register on page 686

15. System control

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

The main functions of the system registers are:

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

The system registers are accessible in AArch64 Execution state at EL0 to EL3. Some of the system registers are accessible through the external debug interface or utility bus interface.

15.1 AArch64 Generic System Control registers

The following summary table provides an overview of all 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.

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 15-1: Generic System Control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ACTLR_EL1	3	0	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL1)
RGSR_EL1	3	0	C1	C0	5	See individual bit resets.	64-bit	Random Allocation Tag Seed Register.
GCR_EL1	3	0	C1	C0	6	See individual bit resets.	64-bit	Tag Control Register.
TTBR0_EL1	3	0	C2	C0	0	See individual bit resets.	64-bit	Translation Table Base Register 0 (EL1)
TTBR1_EL1	3	0	C2	C0	1	See individual bit resets.	64-bit	Translation Table Base Register 1 (EL1)
TCR_EL1	3	0	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL1)
TCR2_EL1	3	0	C2	C0	3	See individual bit resets.	64-bit	Extended Translation Control Register (EL1)
APIAKeyLo_EL1	3	0	C2	C1	0	See individual bit resets.	64-bit	Pointer Authentication Key A for Instruction (bits[63:0])

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
APIAKeyHi_EL1	3	0	C2	C1	1	See individual bit resets.	64-bit	Pointer Authentication Key A for Instruction (bits[127:64])
APIBKeyLo_EL1	3	0	C2	C1	2	See individual bit resets.	64-bit	Pointer Authentication Key B for Instruction (bits[63:0])
APIBKeyHi_EL1	3	0	C2	C1	3	See individual bit resets.	64-bit	Pointer Authentication Key B for Instruction (bits[127:64])
APDAKeyLo_EL1	3	0	C2	C2	0	See individual bit resets.	64-bit	Pointer Authentication Key A for Data (bits[63:0])
APDAKeyHi_EL1	3	0	C2	C2	1	See individual bit resets.	64-bit	Pointer Authentication Key A for Data (bits[127:64])
APDBKeyLo_EL1	3	0	C2	C2	2	See individual bit resets.	64-bit	Pointer Authentication Key B for Data (bits[63:0])
APDBKeyHi_EL1	3	0	C2	C2	3	See individual bit resets.	64-bit	Pointer Authentication Key B for Data (bits[127:64])
APGAKeyLo_EL1	3	0	C2	C3	0	See individual bit resets.	64-bit	Pointer Authentication Key A for Code (bits[63:0])
APGAKeyHi_EL1	3	0	C2	C3	1	See individual bit resets.	64-bit	Pointer Authentication Key A for Code (bits[127:64])
SPSel	3	0	C4	C2	0	See individual bit resets.	64-bit	Stack Pointer Select
CurrentEL	3	0	C4	C2	2	0x000000000000000C	64-bit	Current Exception Level
PAN	3	0	C4	C2	3	See individual bit resets.	64-bit	Privileged Access Never
UAO	3	0	C4	C2	4	See individual bit resets.	64-bit	User Access Override
ALLINT	3	0	C4	C3	0	See individual bit resets.	64-bit	All Interrupt Mask Bit
AFSR0_EL1	3	0	C5	C1	0	See individual bit resets.	64-bit	Auxiliary Fault Status Register 0 (EL1)
AFSR1_EL1	3	0	C5	C1	1	See individual bit resets.	64-bit	Auxiliary Fault Status Register 1 (EL1)
ESR_EL1	3	0	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL1)
TFSR_EL1	3	0	C5	C6	0	See individual bit resets.	64-bit	Tag Fault Status Register (EL1)
TFSRE0_EL1	3	0	C5	C6	1	See individual bit resets.	64-bit	Tag Fault Status Register (EL0).
FAR_EL1	3	0	C6	C0	0	See individual bit resets.	64-bit	Fault Address Register (EL1)
PAR_EL1	3	0	C7	C4	0	See individual bit resets.	64-bit	Physical Address Register
MAIR_EL1	3	0	C10	C2	0	See individual bit resets.	64-bit	Memory Attribute Indirection Register (EL1)
AMAIR_EL1	3	0	C10	C3	0	See individual bit resets.	64-bit	Auxiliary Memory Attribute Indirection Register (EL1)
LORSA_EL1	3	0	C10	C4	0	See individual bit resets.	64-bit	LORegion Start Address (EL1)
LOREA_EL1	3	0	C10	C4	1	See individual bit resets.	64-bit	LORegion End Address (EL1)
LORN_EL1	3	0	C10	C4	2	See individual bit resets.	64-bit	LORegion Number (EL1)
LORC_EL1	3	0	C10	C4	3	See individual bit resets.	64-bit	LORegion Control (EL1)
LORID_EL1	3	0	C10	C4	7	0x0000000000040004	64-bit	LORegionID (EL1)
VBAR_EL1	3	0	C12	C0	0	See individual bit resets.	64-bit	Vector Base Address Register (EL1)
ISR_EL1	3	0	C12	C1	0	See individual bit resets.	64-bit	Interrupt Status Register
CONTEXTIDR_EL1	3	0	C13	C0	1	See individual bit resets.	64-bit	Context ID Register (EL1)
TPIDR_EL1	3	0	C13	C0	4	See individual bit resets.	64-bit	EL1 Software Thread ID Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SCXTNUM_EL1	3	0	C13	C0	7	See individual bit resets.	64-bit	EL1 Read/Write Software Context Number
IMP_CPUACTLR_EL1	3	0	C15	C1	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register (EL1)
IMP_CPUACTLR2_EL1	3	0	C15	C1	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 2 (EL1)
IMP_CPUACTLR3_EL1	3	0	C15	C1	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 3 (EL1)
IMP_CPUACTLR4_EL1	3	0	C15	C1	3	See individual bit resets.	64-bit	CPU Auxiliary Control Register 4 (EL1)
IMP_CPUECTLR_EL1	3	0	C15	C1	4	See individual bit resets.	64-bit	CPU Extended Control Register
IMP_CPUECTLR2_EL1	3	0	C15	C1	5	See individual bit resets.	64-bit	CPU Extended Control Register 2
IMP_CPUL2DIRTYLNCT_EL1	3	0	C15	C2	5	0x0000000000000000	64-bit	CPU L2 Dirty Line Count Register
IMP_CPUSYNCASSISTCR_EL1	3	0	C15	C2	6	See individual bit resets.	64-bit	CPU Synchronization Assist Configuration Register
IMP_CPUPWRCTLR_EL1	3	0	C15	C2	7	See individual bit resets.	64-bit	CPU Power Control Register
IMP_ATCR_EL1	3	0	C15	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register (EL1)
IMP_CPUACTLR5_EL1	3	0	C15	C8	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register 5 (EL1)
IMP_CPUACTLR6_EL1	3	0	C15	C8	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 6 (EL1)
IMP_CPUACTLR7_EL1	3	0	C15	C8	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 7 (EL1)
IMP_CPUACTLR10_EL1	3	0	C15	C8	3	See individual bit resets.	64-bit	CPU Auxiliary Control Register 10 (EL1)
IMP_CPUACTLR8_EL1	3	0	C15	C8	5	See individual bit resets.	64-bit	CPU Auxiliary Control Register 8 (EL1)
IMP_CPUACTLR9_EL1	3	0	C15	C8	6	See individual bit resets.	64-bit	CPU Auxiliary Control Register 9 (EL1)
IMP_CPUACTLR11_EL1	3	0	C15	C8	7	See individual bit resets.	64-bit	CPU Auxiliary Control Register 11 (EL1)
IMP_CPUACTLR12_EL1	3	0	C15	C13	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register 12 (EL1)
IMP_CPUACTLR13_EL1	3	0	C15	C13	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 13 (EL1)
AIDR_EL1	3	1	C0	C0	7	0x0000000000000000	64-bit	Auxiliary ID Register
RNDR	3	3	C2	C4	0	See individual bit resets.	64-bit	Random Number
RNDRRS	3	3	C2	C4	1	See individual bit resets.	64-bit	Random Number Full Entropy
NZCV	3	3	C4	C2	0	See individual bit resets.	64-bit	Condition Flags
DAIF	3	3	C4	C2	1	See individual bit resets.	64-bit	Interrupt Mask Bits
SVCR	3	3	C4	C2	2	See individual bit resets.	64-bit	Streaming Vector Control Register
DIT	3	3	C4	C2	5	See individual bit resets.	64-bit	Data Independent Timing
SSBS	3	3	C4	C2	6	See individual bit resets.	64-bit	Speculative Store Bypass Safe
TCO	3	3	C4	C2	7	See individual bit resets.	64-bit	Tag Check Override
FPCR	3	3	C4	C4	0	See individual bit resets.	64-bit	Floating-point Control Register
FPSR	3	3	C4	C4	1	See individual bit resets.	64-bit	Floating-point Status Register
TPIDR_EL0	3	3	C13	C0	2	See individual bit resets.	64-bit	EL0 Read/Write Software Thread ID Register
TPIDRRO_EL0	3	3	C13	C0	3	See individual bit resets.	64-bit	EL0 Read-Only Software Thread ID Register
TPIDR2_EL0	3	3	C13	C0	5	See individual bit resets.	64-bit	EL0 Read/Write Software Thread ID Register 2
SCXTNUM_EL0	3	3	C13	C0	7	See individual bit resets.	64-bit	EL0 Read/Write Software Context Number

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ACTLR_EL2	3	4	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL2)
HACR_EL2	3	4	C1	C1	7	See individual bit resets.	64-bit	Hypervisor Auxiliary Control Register
TTBR0_EL2	3	4	C2	C0	0	See individual bit resets.	64-bit	Translation Table Base Register 0 (EL2)
TTBR1_EL2	3	4	C2	C0	1	See individual bit resets.	64-bit	Translation Table Base Register 1 (EL2)
TCR_EL2	3	4	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL2)
TCR2_EL2	3	4	C2	C0	3	See individual bit resets.	64-bit	Extended Translation Control Register (EL2)
VTTBR_EL2	3	4	C2	C1	0	See individual bit resets.	64-bit	Virtualization Translation Table Base Register
VTCR_EL2	3	4	C2	C1	2	See individual bit resets.	64-bit	Virtualization Translation Control Register
VSTTBR_EL2	3	4	C2	C6	0	See individual bit resets.	64-bit	Virtualization Secure Translation Table Base Register
VSTCR_EL2	3	4	C2	C6	2	See individual bit resets.	64-bit	Virtualization Secure Translation Control Register
AFSR0_EL2	3	4	C5	C1	0	See individual bit resets.	64-bit	Auxiliary Fault Status Register 0 (EL2)
AFSR1_EL2	3	4	C5	C1	1	See individual bit resets.	64-bit	Auxiliary Fault Status Register 1 (EL2)
ESR_EL2	3	4	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL2)
TFSR_EL2	3	4	C5	C6	0	See individual bit resets.	64-bit	Tag Fault Status Register (EL2)
FAR_EL2	3	4	C6	C0	0	See individual bit resets.	64-bit	Fault Address Register (EL2)
HPFAR_EL2	3	4	C6	C0	4	See individual bit resets.	64-bit	Hypervisor IPA Fault Address Register
MAIR_EL2	3	4	C10	C2	0	See individual bit resets.	64-bit	Memory Attribute Indirection Register (EL2)
AMAIR_EL2	3	4	C10	C3	0	See individual bit resets.	64-bit	Auxiliary Memory Attribute Indirection Register (EL2)
VBAR_EL2	3	4	C12	C0	0	See individual bit resets.	64-bit	Vector Base Address Register (EL2)
CONTEXTIDR_EL2	3	4	C13	C0	1	See individual bit resets.	64-bit	Context ID Register (EL2)
TPIDR_EL2	3	4	C13	C0	2	See individual bit resets.	64-bit	EL2 Software Thread ID Register
SCXTNUM_EL2	3	4	C13	C0	7	See individual bit resets.	64-bit	EL2 Read/Write Software Context Number
IMP_ATCR_EL2	3	4	C15	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register (EL2)
IMP_AVTCR_EL2	3	4	C15	C7	1	See individual bit resets.	64-bit	CPU Virtualization Auxiliary Translation Control Register (EL2)
ACTLR_EL3	3	6	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL3)
SCR_EL3	3	6	C1	C1	0	See individual bit resets.	64-bit	Secure Configuration Register
CPTR_EL3	3	6	C1	C1	2	See individual bit resets.	64-bit	Architectural Feature Trap Register (EL3)
MDCR_EL3	3	6	C1	C3	1	See individual bit resets.	64-bit	Monitor Debug Configuration Register (EL3)
TTBR0_EL3	3	6	C2	C0	0	See individual bit resets.	64-bit	Translation Table Base Register 0 (EL3)
TCR_EL3	3	6	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL3)
AFSR0_EL3	3	6	C5	C1	0	See individual bit resets.	64-bit	Auxiliary Fault Status Register 0 (EL3)
AFSR1_EL3	3	6	C5	C1	1	See individual bit resets.	64-bit	Auxiliary Fault Status Register 1 (EL3)
ESR_EL3	3	6	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL3)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TFSR_EL3	3	6	C5	C6	0	See individual bit resets.	64-bit	Tag Fault Status Register (EL3)
FAR_EL3	3	6	C6	C0	0	See individual bit resets.	64-bit	Fault Address Register (EL3)
MAIR_EL3	3	6	C10	C2	0	See individual bit resets.	64-bit	Memory Attribute Indirection Register (EL3)
AMAIR_EL3	3	6	C10	C3	0	See individual bit resets.	64-bit	Auxiliary Memory Attribute Indirection Register (EL3)
VBAR_EL3	3	6	C12	C0	0	See individual bit resets.	64-bit	Vector Base Address Register (EL3)
RVBAR_EL3	3	6	C12	C0	1	See individual bit resets.	64-bit	Reset Vector Base Address Register (if EL3 implemented)
RMR_EL3	3	6	C12	C0	2	See individual bit resets.	64-bit	Reset Management Register (EL3)
TPIDR_EL3	3	6	C13	C0	2	See individual bit resets.	64-bit	EL3 Software Thread ID Register
SCXTNUM_EL3	3	6	C13	C0	7	See individual bit resets.	64-bit	EL3 Read/Write Software Context Number
IMP_CPUL2SDIRTYLNCT_EL3	3	6	C15	C2	3	0x0000000000000000	64-bit	CPU L2 Secure Dirty Line Count Register
IMP_CPUACTLR_EL3	3	6	C15	C4	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register (EL3)
IMP_ATCR_EL3	3	6	C15	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register (EL3)
IMP_CPUPSELR_EL3	3	6	C15	C8	0	See individual bit resets.	64-bit	Selected Instruction Private Select Register
IMP_CPUPCR_EL3	3	6	C15	C8	1	See individual bit resets.	64-bit	Selected Instruction Private Control Register
IMP_CPUPOR_EL3	3	6	C15	C8	2	See individual bit resets.	64-bit	Selected Instruction Private Opcode Register
IMP_CPUPMR_EL3	3	6	C15	C8	3	See individual bit resets.	64-bit	Selected Instruction Private Mask Register
IMP_CPUPOR2_EL3	3	6	C15	C8	4	See individual bit resets.	64-bit	Selected Instruction Private Opcode Register 2
IMP_CPUPMR2_EL3	3	6	C15	C8	5	See individual bit resets.	64-bit	Selected Instruction Private Mask Register 2
IMP_CPUPFR_EL3	3	6	C15	C8	6	See individual bit resets.	64-bit	Selected Instruction Private Flag Register

16. Debug

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.

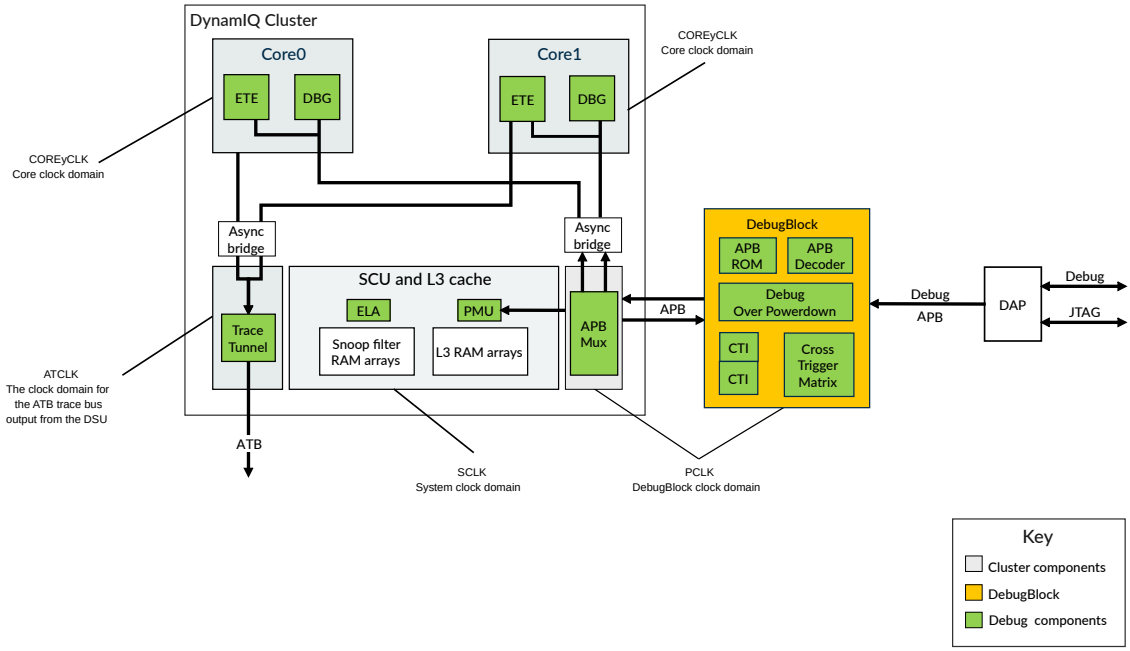
The connection between the cluster and the DebugBlock consists of a pair of Advanced Peripheral Bus (APB) interfaces, one in each direction. All debug traffic, except the authentication interface, takes place over this interface as read or write APB transactions. This debug traffic includes register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

Figure 16-1: C1-DSU cluster debug components



The primary debug APB interface on the DebugBlock controls the debug components. The APB decoder decodes the requests on this bus before they are sent to the appropriate component in the DebugBlock or in the C1-DSU cluster. The per-core CTIs are connected to a CTM.

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

The trace unit in each core outputs trace, which is funneled in the C1-DSU cluster down to a single AMBA® 4 ATBv1.1 interface.

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

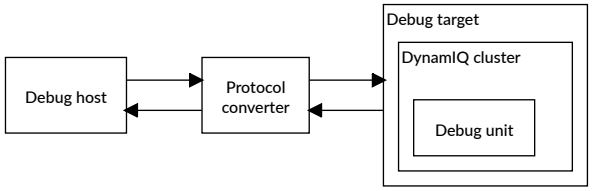
The C1-Ultra core also supports direct access to internal memory, also known as cache debug. Direct access to internal memory allows software to read the internal memory that the L1 and L2 cache and Translation Lookaside Buffer (TLB) structures use. For more information about cache debug and direct access to internal memory, see 9. Direct access to internal memory on page 76.

16.1 Supported debug methods

The C1-DSU cluster along with its associated cores is part of a debug system that supports both self-hosted and external debug.

The following figure shows a typical external debug system.

Figure 16-2: External debug system



Debug host

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

Protocol converter

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

Debug target

The lowest level of the system implements system support for the protocol converter to access the debug unit. For C1-DSU based devices, the mechanism used to access the debug unit is based on the CoreSight architecture. The C1-DSU DebugBlock is accessed using an Advanced Peripheral Bus (APB) interface and the debug accesses are then directed to the selected C1-Ultra core inside the C1-DSU cluster. An example of a debug target is a development system with a test chip or a silicon part with a C1-Ultra core.

Debug unit

Helps debugging software that is running on the core:

- C1-DSU and external hardware based around the core
- Operating systems
- Application software

With the debug unit, you can:

- Stop program execution
- Examine and alter process and coprocessor state
- Examine and alter memory and the state of the input or output peripherals
- Restart the core

For self-hosted debug, the debug target runs debug monitor software that runs on the core in the C1-DSU cluster. This way, it does not require expensive interface hardware to connect a second host computer.

16.2 Debug register interfaces

The C1-Ultra core implements the Arm®v8.8 debug architecture and supports Arm®v8.3-A Debug over Powerdown (DoPD).

The Debug architecture defines a set of Debug registers. The Debug register interfaces provide access to these registers either from software running on the core or from an external debugger. For more information, see *Debug* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

Related information

4.7 Debug over powerdown on page 55

16.2.1 Core interfaces

System register access allows the C1-Ultra core to access certain Debug registers directly. The Debug register interfaces provide access to these registers either from software running on the core or from an external debugger.

Access to the Debug registers is partitioned as follows:

Debug

This function is both system register based and memory-mapped. You can access the Debug register map using the APB completer port that connects into the DebugBlock of the C1-DynamIQ™ Shared Unit (DSU).

Performance monitoring

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

Trace

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

Statistical profiling

This function is system register based.

ELA registers

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

The ELA-600 is licensed separately.


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*.

16.2.2 Breakpoints and watchpoints

The C1-Ultra core supports six breakpoints, four watchpoints, and a standard Debug Communications Channel (DCC).

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

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

16.3 Debug events

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

The C1-Ultra core responds to a debug event in one of the following ways:

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

In the C1-Ultra core, watchpoint debug events are always synchronous. Memory hint instructions and cache clean operations, except `DC ZVA`, and `DC IVAC` do not generate watchpoint debug events. Store exclusive instructions generate a watchpoint debug event even when the check for the control of exclusive monitor fails. Atomic `CAS` instructions generate a watchpoint debug event even when the compare operation fails.

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

16.4 Debug memory map and debug signals

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

For more information about debug memory maps and debug signals in a C1-DSU cluster, see the *Debug* and *ROM tables* chapters in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

16.5 ROM table

The C1-Ultra core 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 System on Chip (SoC) and is for the C1-Ultra core. There is one ROM table for each core and each 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 and each C1-SME2 unit. For more information, see *ROM tables* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

The C1-Ultra core ROM table includes the following entries:

Table 16-1: Core ROM table

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

Related information

B.3 External CoreROM registers summary on page 657

16.6 CoreSight component identification

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

Table 16-2: C1-Ultra core CoreSight™ component identification

Component	Peripheral ID	Component ID	DevType	DevArch	Core revision
Debug	0x04001BBD8C	0xB105900D	0x15	0x4770AA15	r1p0
PMU			0x16	0x47702A16	

Component	Peripheral ID	Component ID	DevType	DevArch	Core revision
Trace unit			0x13	0x47715A13	
ROM table			0x00	0x47700AF7	

16.7 AArch64 Debug registers

The following summary table provides an overview of all Debug registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 16-3: Debug registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
OSDTRRX_EL1	2	0	C0	C0	2	See individual bit resets.	64-bit	OS Lock Data Transfer Register, Receive
DBGBVR0_EL1	2	0	C0	C0	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR0_EL1	2	0	C0	C0	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR0_EL1	2	0	C0	C0	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR0_EL1	2	0	C0	C0	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
DBGBVR1_EL1	2	0	C0	C1	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR1_EL1	2	0	C0	C1	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR1_EL1	2	0	C0	C1	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR1_EL1	2	0	C0	C1	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
MDCCINT_EL1	2	0	C0	C2	0	See individual bit resets.	64-bit	Monitor DCC Interrupt Enable Register
MDSCR_EL1	2	0	C0	C2	2	See individual bit resets.	64-bit	Monitor Debug System Control Register
DBGBVR2_EL1	2	0	C0	C2	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR2_EL1	2	0	C0	C2	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR2_EL1	2	0	C0	C2	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR2_EL1	2	0	C0	C2	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
OSDTRTX_EL1	2	0	C0	C3	2	See individual bit resets.	64-bit	OS Lock Data Transfer Register, Transmit
DBGBVR3_EL1	2	0	C0	C3	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR3_EL1	2	0	C0	C3	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR3_EL1	2	0	C0	C3	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR3_EL1	2	0	C0	C3	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
DBGBVR4_EL1	2	0	C0	C4	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR4_EL1	2	0	C0	C4	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGBVR5_EL1	2	0	C0	C5	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
DBGBCR5_EL1	2	0	C0	C5	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
OSECCR_EL1	2	0	C0	C6	2	See individual bit resets.	64-bit	OS Lock Exception Catch Control Register
MDRAR_EL1	2	0	C1	C0	0	See individual bit resets.	64-bit	Monitor Debug ROM Address Register
OSLAR_EL1	2	0	C1	C0	4	See individual bit resets.	64-bit	OS Lock Access Register
OSLSR_EL1	2	0	C1	C1	4	See individual bit resets.	64-bit	OS Lock Status Register
OSDLR_EL1	2	0	C1	C3	4	See individual bit resets.	64-bit	OS Double Lock Register
DBGPRCR_EL1	2	0	C1	C4	4	See individual bit resets.	64-bit	Debug Power Control Register
DBGCLAIMSET_EL1	2	0	C7	C8	6	See individual bit resets.	64-bit	Debug CLAIM Tag Set Register
DBGCLAIMCLR_EL1	2	0	C7	C9	6	See individual bit resets.	64-bit	Debug CLAIM Tag Clear Register
DBGAUTHSTATUS_EL1	2	0	C7	C14	6	See individual bit resets.	64-bit	Debug Authentication Status Register
MDCCSR_EL0	2	3	C0	C1	0	See individual bit resets.	64-bit	Monitor DCC Status Register
DBGDTR_EL0	2	3	C0	C4	0	See individual bit resets.	64-bit	Debug Data Transfer Register, half-duplex
DBGDTRRX_EL0	2	3	C0	C5	0	See individual bit resets.	64-bit	Debug Data Transfer Register, Receive
DBGDTRTX_EL0	2	3	C0	C5	0	See individual bit resets.	64-bit	Debug Data Transfer Register, Transmit
TRFCR_EL1	3	0	C1	C2	1	See individual bit resets.	64-bit	Trace Filter Control Register (EL1)
MDCR_EL2	3	4	C1	C1	1	See individual bit resets.	64-bit	Monitor Debug Configuration Register (EL2)
TRFCR_EL2	3	4	C1	C2	1	See individual bit resets.	64-bit	Trace Filter Control Register (EL2)
IMP_IDATA0_EL3	3	6	C15	C0	0	0x0000000000000000	64-bit	Instruction Register 0
IMP_IDATA1_EL3	3	6	C15	C0	1	0x0000000000000000	64-bit	Instruction Register 1
IMP_IDATA2_EL3	3	6	C15	C0	2	0x0000000000000000	64-bit	Instruction Register 2
IMP_DDATA0_EL3	3	6	C15	C1	0	0x0000000000000000	64-bit	Data Register 0
IMP_DDATA1_EL3	3	6	C15	C1	1	0x0000000000000000	64-bit	Data Register 1
IMP_DDATA2_EL3	3	6	C15	C1	2	0x0000000000000000	64-bit	Data Register 2

16.8 External Debug registers

The following summary table provides an overview of all memory-mapped Debug registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 16-4: Debug registers summary

Offset	Name	Reset	Width	Description
0x020	EDESR	See individual bit resets.	32-bit	External Debug Event Status Register

Offset	Name	Reset	Width	Description
0x024	EDECR	See individual bit resets.	32-bit	External Debug Execution Control Register
0x030	EDWAR	See individual bit resets.	64-bit	External Debug Watchpoint Address Register
0x038	EDHSR	See individual bit resets.	64-bit	External Debug Halting Syndrome Register
0x080	DBGDTRRX_EL0	See individual bit resets.	32-bit	Debug Data Transfer Register, Receive
0x084	EDITR	See individual bit resets.	32-bit	External Debug Instruction Transfer Register
0x088	EDSCR	See individual bit resets.	32-bit	External Debug Status and Control Register
0x08C	DBGDTRTX_EL0	See individual bit resets.	32-bit	Debug Data Transfer Register, Transmit
0x090	EDRCR	See individual bit resets.	32-bit	External Debug Reserve Control Register
0x098	EDECCR	See individual bit resets.	32-bit	External Debug Exception Catch Control Register
0x300	OSLAR_EL1	See individual bit resets.	32-bit	OS Lock Access Register
0x310	EDPRCR	See individual bit resets.	32-bit	External Debug Power/Reset Control Register
0x314	EDPRSR	See individual bit resets.	32-bit	External Debug Processor Status Register
0x400	DBGBVR0_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x408	DBGBCR0_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x410	DBGBVR1_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x418	DBGBCR1_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x420	DBGBVR2_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x428	DBGBCR2_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x430	DBGBVR3_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x438	DBGBCR3_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x440	DBGBVR4_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x448	DBGBCR4_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x450	DBGBVR5_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x458	DBGBCR5_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x800	DBGWVR0_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x808	DBGWCR0_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x810	DBGWVR1_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x818	DBGWCR1_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x820	DBGWVR2_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x828	DBGWCR2_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x830	DBGWVR3_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x838	DBGWCR3_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0xD00	MIDR_EL1	0x411FD8C0	32-bit	Main ID Register
0xD20	EDPFR	See individual bit resets.	64-bit	External Debug Processor Feature Register
0xD28	EDDFR	See individual bit resets.	64-bit	External Debug Feature Register
0xD48	EDDFR1	See individual bit resets.	64-bit	External Debug Feature Register 1
0xD60	EDAA32PFR	0x0000000000000000	64-bit	External Debug Auxiliary Processor Feature Register
0xF00	EDITCTRL	See individual bit resets.	32-bit	External Debug Integration mode Control register
0xFA0	DBGCLAIMSET_EL1	See individual bit resets.	32-bit	Debug CLAIM Tag Set Register
0xFA4	DBGCLAIMCLR_EL1	0x00000000	32-bit	Debug CLAIM Tag Clear Register

Offset	Name	Reset	Width	Description
0xFA8	EDDEVAFF0	See individual bit resets.	32-bit	External Debug Device Affinity register 0
0xFAC	EDDEVAFF1	See individual bit resets.	32-bit	External Debug Device Affinity register 1
0xFB0	EDLAR	See individual bit resets.	32-bit	External Debug Lock Access Register
0xFB4	EDLSR	See individual bit resets.	32-bit	External Debug Lock Status Register
0xFB8	DBGAUTHSTATUS_EL1	See individual bit resets.	32-bit	Debug Authentication Status Register
0xFBC	EDDEVARCH	0x4770AA15	32-bit	External Debug Device Architecture Register
0xFC0	EDDEVID2	0x00000000	32-bit	External Debug Device ID register 2
0xFC4	EDDEVID1	0x00000010	32-bit	External Debug Device ID Register 1
0xFC8	EDDEVID	0x00000010	32-bit	External Debug Device ID register 0
0xFCC	EDDEVTYPE	0x00000015	32-bit	External Debug Device Type register
0xFD0	EDPIDR4	0x00000004	32-bit	External Debug Peripheral Identification Register 4
0xFE0	EDPIDR0	0x0000008C	32-bit	External Debug Peripheral Identification Register 0
0xFE4	EDPIDR1	0x000000BD	32-bit	External Debug Peripheral Identification Register 1
0xFE8	EDPIDR2	0x0000001B	32-bit	External Debug Peripheral Identification Register 2
0xFEC	EDPIDR3	0x00000000	32-bit	External Debug Peripheral Identification Register 3
0xFF0	EDCIDR0	0x0000000D	32-bit	External Debug Component Identification Register 0
0xFF4	EDCIDR1	0x00000090	32-bit	External Debug Component Identification Register 1
0xFF8	EDCIDR2	0x00000005	32-bit	External Debug Component Identification Register 2
0xFFC	EDCIDR3	0x000000B1	32-bit	External Debug Component Identification Register 3

16.9 External CoreROM registers

The following summary table provides an overview of all memory-mapped CoreROM registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 16-5: CoreROM registers summary

Offset	Name	Reset	Width	Description
0x000	COREROM_ROMENTRY0	0x00010003	32-bit	Core ROM table Entry 0
0x004	COREROM_ROMENTRY1	0x00020003	32-bit	Core ROM table Entry 1
0x008	COREROM_ROMENTRY2	0x00030003	32-bit	Core ROM table Entry 2
0x00C	COREROM_ROMENTRY3	0x00040003	32-bit	Core ROM table Entry 3
0xFB8	COREROM_AUTHSTATUS	0x00000000	32-bit	Core ROM table Authentication Status Register

Offset	Name	Reset	Width	Description
0xFBC	COREROM_DEVARCH	0x47700AF7	32-bit	Core ROM table Device Architecture Register
0xFCC	COREROM_DEVTYPE	0x00000000	32-bit	Core ROM table Device Type Register
0xFD0	COREROM_PIDR4	0x00000004	32-bit	Core ROM table Peripheral Identification Register 4
0xFE0	COREROM_PIDR0	0x0000008C	32-bit	Core ROM table Peripheral Identification Register 0
0xFE4	COREROM_PIDR1	0x000000BD	32-bit	Core ROM table Peripheral Identification Register 1
0xFE8	COREROM_PIDR2	0x0000001B	32-bit	Core ROM table Peripheral Identification Register 2
0xFEC	COREROM_PIDR3	0x00000000	32-bit	Core ROM table Peripheral Identification Register 3
0xFF0	COREROM_CIDR0	0x0000000D	32-bit	Core ROM table Component Identification Register 0
0xFF4	COREROM_CIDR1	0x00000090	32-bit	Core ROM table Component Identification Register 1
0xFF8	COREROM_CIDR2	0x00000005	32-bit	Core ROM table Component Identification Register 2
0xFFC	COREROM_CIDR3	0x000000B1	32-bit	Core ROM table Component Identification Register 3

17. Performance Monitors Extension support

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

The C1-Ultra core Performance Monitoring Unit (PMU):

- Collects events through an event interface from other units in the design. These events are used as triggers for event counters.
- Supports cycle counting through the Performance Monitors Control Register.
- Implements PMU snapshots for context samples.
- Provides 6 or 31 64-bit PMU counters that count any of the events available in the core. The absolute counts that are recorded might vary because of pipeline effects. This variation has little effect except in cases where the counters are enabled for a very short time.

You can program the PMU using either the System registers or the external Debug Advanced Peripheral Bus (APB) interface.

17.1 Common PMU events

The C1-Ultra core 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-Ultra core performance monitoring unit (PMU) 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 these 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*.

Table 17-1: Common PMU events

Event number	Mnemonic	Description
0x0000	SW_INCR	**Instruction architecturally executed, Condition code check pass, software increment** Counts software writes to the PMSWINC_EL0 (software PMU increment) register. The PMSWINC_EL0 register is a manually updated counter for use by application software. This event could be used to measure any user program event, such as accesses to a particular data structure (by writing to the PMSWINC_EL0 register each time the data structure is accessed). To use the PMSWINC_EL0 register and event, developers must insert instructions that write to the PMSWINC_EL0 register into the source code. Since the SW_INCR event records writes to the PMSWINC_EL0 register, there is no need to do a read/increment/write sequence to the PMSWINC_EL0 register. **Note:** This event is not exported to the trace unit.
0x0001	L1I_CACHE_REFILL	**Level 1 instruction cache refill** Counts cache line refills in the level 1 instruction cache caused by a missed instruction fetch. Instruction fetches may include accessing multiple instructions, but the single cache line allocation is counted once.
0x0002	L1I_TLB_REFILL	**Level 1 instruction TLB refill** Counts level 1 instruction TLB refills from any Instruction fetch. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event will not count if the translation table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB.
0x0003	L1D_CACHE_REFILL	**Level 1 data cache refill** Counts level 1 data cache refills caused by speculatively executed load or store operations that missed in the level 1 data cache. This event only counts one event per cache line. Since the caches are write-back only for this processor, there are no write-through cache accesses.
0x0004	L1D_CACHE	**Level 1 data cache access** Counts level 1 data cache accesses from any load/store operations. Atomic operations that resolve in the CPUs caches (near atomic operations) counts as both a write access and read access. Each access to a cache line is counted including the multiple accesses caused by single instructions such as LDM or STM. Each access to other level 1 data or unified memory structures, for example refill buffers, write buffers, and write-back buffers, are also counted.

Event number	Mnemonic	Description
0x0005	L1D_TLB_REFILL	**Level 1 data TLB refill** Counts level 1 data TLB accesses that resulted in TLB refills. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event counts for refills caused by preload instructions or hardware prefetch accesses. This event counts regardless of whether the miss hits in L2 or results in a translation table walk. This event will not count if the translation table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB. This event will not count on an access from an AT(address translation) instruction. This event is the sum of the L1D_TLB_REFILL_RD and L1D_TLB_REFILL_WR events.
0x0008	INST_RETIRED	**Instruction architecturally executed** Counts instructions that have been architecturally executed.
0x0009	EXC_TAKEN	**Exception taken** Counts any taken architecturally visible exceptions such as IRQ, FIQ, SError, and other synchronous exceptions. Exceptions are counted whether or not they are taken locally.
0x000A	EXC_RETURN	**Instruction architecturally executed, Condition code check pass, exception return** Counts any architecturally executed exception return instructions. For example: AArch64: ERET
0x000B	CID_WRITE_RETIRED	**Instruction architecturally executed, Condition code check pass, write to CONTEXTIDR** Counts architecturally executed writes to the CONTEXTIDR_EL1 register, which usually contain the kernel PID and can be output with hardware trace.
0x000C	PC_WRITE_RETIRED	**Instruction architecturally executed, Condition code check pass, Software change of the PC** Counts branch instructions that caused a change of Program Counter, which effectively causes a change in the control flow of the program.
0x000D	BR_IMMED_RETIRED	**Branch instruction architecturally executed, immediate** Counts architecturally executed direct branches. **Note:** This event is not exported to the trace unit.
0x000E	BR_RETURN_RETIRED	**Branch instruction architecturally executed, procedure return, taken** Counts architecturally executed procedure returns. **Note:** This event is not exported to the trace unit.
0x0010	BR_MIS_PRED	**Branch instruction speculatively executed, mispredicted or not predicted** Counts branches which are speculatively executed and mispredicted.
0x0011	CPU_CYCLES	**Cycle** Counts CPU clock cycles (not timer cycles). The clock measured by this event is defined as the physical clock driving the CPU logic.

Event number	Mnemonic	Description
0x0012	BR_PRED	**Predictable branch instruction speculatively executed** Counts all speculatively executed branches.
0x0013	MEM_ACCESS	**Data memory access** Counts memory accesses issued by the CPU load store 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. Memory accesses generated by the following instructions or activity are not counted: Instruction fetches, Cache maintenance instructions, Translation table walks or prefetches, Memory prefetch operations. This event counts the sum of the MEM_ACCESS_RD and MEM_ACCESS_WR events.
0x0014	L1I_CACHE	**Level 1 instruction cache access** Counts instruction fetches which access the level 1 instruction cache. Instruction cache accesses caused by cache maintenance operations are not counted.
0x0015	L1D_CACHE_WB	**Level 1 data cache write-back** Counts write-backs of dirty data from the L1 data cache to the L2 cache. This occurs when either a dirty cache line is evicted from L1 data cache and allocated in the L2 cache or dirty data is written to the L2 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 L2 without writing L1, such as write streaming mode. This event is the sum of the L1D_CACHE_WB_CLEAN and L1D_CACHE_WB_VICTIM events.
0x0016	L2D_CACHE	**Level 2 data cache access** Counts accesses to the level 2 cache due to data accesses. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the first level data cache or translation resolutions due to accesses. This event also counts write back of dirty data from level 1 data cache to the L2 cache. This event is the sum of the L2D_CACHE_RD, L2D_CACHE_WR, L2D_CACHE_PRFM, IMP_L2D_CACHE_L1HWPRF, and L2D_CACHE_HWPRF events.
0x0017	L2D_CACHE_REFILL	**Level 2 data cache refill** Counts cache line refills into the level 2 cache. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This event is the sum of L2D_CACHE_REFILL_RD, L2D_CACHE_REFILL_WR, IMP_L2D_CACHE_REFILL_L1HWPRF, L2D_CACHE_REFILL_HWPRF, and L2D_CACHE_REFILL_PRFM.

Event number	Mnemonic	Description
0x0018	L2D_CACHE_WB	**Level 2 data cache write-back** Counts write-backs of data from the L2 cache to outside the CPU. This includes snoops to the L2 (from other CPUs) which return data even if the snoops cause an invalidation. L2 cache line invalidations which do not write data outside the CPU and snoops which return data from an L1 cache are not counted. Data would not be written outside the cache when invalidating a clean cache line. This event is the sum of the L2D_CACHE_WB_VICTIM and L2D_CACHE_WB_CLEAN events.
0x0019	BUS_ACCESS	**Bus access** Counts memory transactions issued by the CPU to the external bus, including snoop requests and snoop responses. Each beat of data is counted individually.
0x001A	MEMORY_ERROR	**Local memory error** Counts any detected correctable or uncorrectable physical memory errors (ECC or parity) in protected CPUs RAMs. On the core, this event counts errors in the caches (including data and tag rams). Any detected memory error (from either a speculative and abandoned access, or an architecturally executed access) is counted. Note that errors are only detected when the actual protected memory is accessed by an operation.
0x001B	INST_SPEC	**Operation speculatively executed** Counts operations that have been speculatively executed.
0x001C	TTBR_WRITE_RETIRED	**Instruction architecturally executed, Condition code check pass, write to TTBR** Counts architectural writes to TTBR0/1_EL1. If virtualization host extensions are enabled (by setting the HCR_EL2.E2H bit to 1), then accesses to TTBR0/1_EL1 that are redirected to TTBR0/1_EL2, or accesses to TTBR0/1_EL12, are counted. TTBRn registers are typically updated when the kernel is swapping user-space threads or applications.
0x001D	BUS_CYCLES	**Bus cycle** Counts bus cycles in the CPU. Bus cycles represent a clock cycle in which a transaction could be sent or received on the interface from the CPU to the external bus. Since that interface is driven at the same clock speed as the CPU, this event is a duplicate of CPU_CYCLES.
0x001E	CHAIN	**Chain a pair of event counters** For odd-numbered counters, this event increments the count by one for each overflow of the preceding even-numbered counter. For even-numbered counters, there is no increment. This event is used when the even/odd pairs of registers are used as a single counter.
0x0020	L2D_CACHE_ALLOCATE	**Level 2 data cache allocation without refill** Counts level 2 cache line allocates that do not fetch data from outside the level 2 data or unified cache.
0x0021	BR_RETIRED	**Instruction architecturally executed, branch** Counts architecturally executed branches, whether the branch is taken or not. Instructions that explicitly write to the PC are also counted. Note that exception generating instructions, exception return instructions and context synchronization instructions are not counted.
0x0022	BR_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted** Counts branches counted by BR_RETIRED which were mispredicted and caused a pipeline flush.

Event number	Mnemonic	Description
0x0023	STALL_FRONTEND	**No operation sent for execution due to the frontend** Counts cycles when frontend could not send any micro-operations to the rename stage because of frontend resource stalls caused by fetch memory latency or branch prediction flow stalls. STALL_FRONTEND_SLOTS counts SLOTS during the cycle when this event counts. STALL_SLOT_FRONTEND will count SLOTS when this event is counted on this CPU.
0x0024	STALL_BACKEND	**No operation sent for execution due to the backend** Counts cycles whenever the rename 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. All the backend slots were empty during the cycle when this event counts. STALL_SLOT_BACKEND will count SLOTS when this event is counted on this CPU.
0x0025	L1D_TLB	**Level 1 data TLB access** Counts level 1 data TLB accesses caused by any memory load or store operation. Note that load or store instructions can be broken up into multiple memory operations. This event does not count TLB maintenance operations.
0x0026	L1I_TLB	**Level 1 instruction TLB access** Counts level 1 instruction TLB accesses, whether the access hits or misses in the TLB. This event counts both demand accesses and prefetch or preload generated accesses. This event is a superset of the L1I_TLB_REFILL event.
0x0027	L2I_CACHE	**Level 2 instruction cache access** Counts accesses to the level 2 cache due to instruction accesses. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the first level instruction cache. **Note:** This event is not exported to the trace unit.
0x0028	L2I_CACHE_REFILL	**Level 2 instruction cache refill** Counts cache line refills into the level 2 cache. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the level 1 instruction cache. **Note:** This event is not exported to the trace unit.
0x0029	L3D_CACHE_ALLOCATE	**Level 3 data cache allocation 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.
0x002A	L3D_CACHE_REFILL	**Level 3 data cache refill** Counts level 3 accesses that receive data from outside the L3 cache.

Event number	Mnemonic	Description
0x002B	L3D_CACHE	**Level 3 data cache access** 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.
0x002D	L2D_TLB_REFILL	**Level 2 data TLB refill** Counts level 2 TLB refills caused by memory operations from both data and instruction fetch, except for those caused by TLB maintenance operations and hardware prefetches.
0x002F	L2D_TLB	**Level 2 data TLB access** Counts level 2 TLB accesses except those caused by TLB maintenance operations. This event is the sum of the L2D_TLB_RD and L2D_TLB_WR events.
0x0031	REMOTE_ACCESS	**Access to another socket in a multi-socket system** 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.
0x0032	LL_CACHE	**Last level cache access** Counts transactions that were returned from outside the core cluster. This event counts transactions for external last level cache when the system register CPUECTLR.EXTLLC bit is set. Counts the same as LL_CACHE_RD on this CPU. **Note:** This event is not exported to the trace unit.
0x0034	DTLB_WALK	**Data TLB access with at least one translation table walk** Counts number of demand data translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations. This event does not include prefetches.
0x0035	ITLB_WALK	**Instruction TLB access with at least one translation table walk** Counts number of instruction translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations. This event does not include prefetches.

Event number	Mnemonic	Description
0x0036	LL_CACHE_RD	**Last level cache access, read** Counts read transactions that were returned from outside the core cluster. This event counts for external last level cache when the system register CPUECTLR.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 LL_CACHE_MISS_RD event.
0x0037	LL_CACHE_MISS_RD	**Last level cache miss, read** Counts read transactions that were returned from outside the core cluster but missed in the system level cache. This event counts for external last level cache when the system register CPUECTLR.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 LL_CACHE_RD event.
0x0039	L1D_CACHE_LMISS_RD	**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. Counts same as L1D_CACHE_REFILL_RD on this CPU.
0x003A	OP_RETIRED	**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.
0x003B	OP_SPEC	**Micro-operation speculatively executed** Counts micro-operations speculatively executed. This is the count of the number of micro-operations dispatched in a cycle.
0x003C	STALL	**No operation sent for execution** Counts cycles when no operations are sent to the rename unit from the frontend or from the rename unit to the backend for any reason (either frontend or backend stall). This event is the sum of STALL_FRONTEND and STALL_BACKEND STALL is a sum of STALL_FRONTEND and STALL_BACKEND on this CPU
0x003D	STALL_SLOT_BACKEND	**No operation sent for execution on a Slot due to the backend** Counts slots per cycle in which no operations are sent from the rename unit to the backend due to backend resource constraints. STALL_BACKEND counts during the cycle when STALL_SLOT_BACKEND counts at least 1. STALL_BACKEND counts during the cycle when STALL_SLOT_BACKEND is SLOTS
0x003E	STALL_SLOT_FRONTEND	**No operation sent for execution on a Slot due to the frontend** Counts slots per cycle in which no operations are sent to the rename unit from the frontend due to frontend resource constraints. STALL_FRONTEND counts during the cycle when STALL_SLOT_FRONTEND is SLOTS

Event number	Mnemonic	Description
0x003F	STALL_SLOT	**No operation sent for execution on a Slot** Counts slots per cycle in which no operations are sent to the rename unit from the frontend or from the rename unit to the backend for any reason (either frontend or backend stall). STALL_SLOT is the sum of STALL_SLOT_FRONTEND and STALL_SLOT_BACKEND. STALL_SLOT is a sum of STALL_SLOT_FRONTEND and STALL_SLOT_BACKEND on this CPU
0x0040	L1D_CACHE_RD	**Level 1 data cache access, read** Counts level 1 data cache accesses from any load operation. Atomic load operations that resolve in the CPUs caches counts as both a write access and read access.
0x0041	L1D_CACHE_WR	**Level 1 data cache access, write** Counts level 1 data cache accesses generated by store operations. This event also counts accesses caused by a DC ZVA (data cache zero, specified by virtual address) instruction. Near atomic operations that resolve in the CPUs caches count as a write access and read access. This event is a subset of the L1D_CACHE event, except this event only counts memory-write operations.
0x0042	L1D_CACHE_REFILL_RD	**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. This event is a subset of the L1D_CACHE_REFILL event, but only counts memory read operations. This event does not count reads caused by cache maintenance operations or preload instructions.
0x0043	L1D_CACHE_REFILL_WR	**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. This event is a subset of the L1D_CACHE_REFILL event, but only counts memory write operations.
0x0044	L1D_CACHE_REFILL_INNER	**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.
0x0045	L1D_CACHE_REFILL_OUTER	**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.
0x0046	L1D_CACHE_WB_VICTIM	**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. This event is a subset of the L1D_CACHE_WB event, but the event only counts write-backs that are a result of the line being allocated for an access made by the CPU.

Event number	Mnemonic	Description
0x0047	L1D_CACHE_WB_CLEAN	**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. This event is a subset of the L1D_CACHE_WB event.
0x0048	L1D_CACHE_INVAL	**Level 1 data cache invalidate** Counts each explicit invalidation of a cache line in the level 1 data cache caused by: • Cache Maintenance Operations (CMO) that operate by a virtual address. • 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. A CMO which is executed on that CPU and invalidates a cache line specified by set/way. Note that CMOs that operate by set/way cannot be broadcast from one CPU to another.
0x004C	L1D_TLB_REFILL_RD	**Level 1 data TLB refill, read** Counts level 1 data TLB refills caused by memory read operations. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event counts for refills caused by preload instructions or hardware prefetch accesses. This event counts regardless of whether the miss hits in L2 or results in a translation table walk. This event will not count if the translation table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB. This event will not count on an access from an Address Translation (AT) instruction. This event is a subset of the L1D_TLB_REFILL event.
0x004D	L1D_TLB_REFILL_WR	**Level 1 data TLB refill, write** Counts level 1 data TLB refills caused by data side memory write operations. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event counts for refills caused by preload instructions or hardware prefetch accesses. This event counts regardless of whether the miss hits in L2 or results in a translation table walk. This event will not count if the table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB. This event will not count with an access from an Address Translation (AT) instruction. This event is a subset of the L1D_TLB_REFILL event.
0x004E	L1D_TLB_RD	**Level 1 data TLB access, read** Counts level 1 data TLB accesses caused by memory read operations. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.
0x004F	L1D_TLB_WR	**Level 1 data TLB access, write** Counts any L1 data side TLB accesses caused by memory write operations. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.

Event number	Mnemonic	Description
0x0050	L2D_CACHE_RD	**Level 2 data cache access, read** Counts level 2 data cache accesses due to memory read operations. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This event is a subset of the L2D_CACHE event, but this event only counts memory read operations.
0x0051	L2D_CACHE_WR	**Level 2 data cache access, write** Counts level 2 cache accesses due to memory write operations. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This event is a subset of the L2D_CACHE event, but this event only counts memory write operations.
0x0052	L2D_CACHE_REFILL_RD	**Level 2 data cache refill, read** Counts refills for memory accesses due to memory read operation counted by L2D_CACHE_RD. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This event is a subset of the L2D_CACHE_REFILL event. This event does not count L2 refills caused by stashes into L2.
0x0053	L2D_CACHE_REFILL_WR	**Level 2 data cache refill, write** Counts refills for memory accesses due to memory write operation counted by L2D_CACHE_WR. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This event does not count on this CPU
0x0056	L2D_CACHE_WB_VICTIM	**Level 2 data cache write-back, victim** Counts evictions from the level 2 cache because of a line being allocated into the L2 cache. This event is a subset of the L2D_CACHE_WB event.
0x0057	L2D_CACHE_WB_CLEAN	**Level 2 data cache write-back, cleaning and coherency** Counts write-backs from the level 2 cache that are a result of either: 1. Cache maintenance operations, 2. Snoop responses or, 3. Direct cache transfers to another CPU due to a forwarding snoop request. This event is a subset of the L2D_CACHE_WB event.

Event number	Mnemonic	Description
0x0058	L2D_CACHE_INVAL	**Level 2 data cache invalidate** Counts each explicit invalidation of a cache line in the level 2 cache by cache maintenance operations that operate by a virtual address, or by external coherency operations. This event does not count if either: 1. A cache refill invalidates a cache line or, 2. A Cache Maintenance Operation (CMO), which invalidates a cache line specified by set/way, is executed on that CPU. CMOs that operate by set/way cannot be broadcast from one CPU to another.
0x005C	L2D_TLB_REFILL_RD	**Level 2 data TLB refill, read** Counts level 2 TLB refills caused by memory read operations from both data and instruction fetch except for those caused by TLB maintenance operations or hardware prefetches. This event is a subset of the L2D_TLB_REFILL event.
0x005D	L2D_TLB_REFILL_WR	**Level 2 data TLB refill, write** Counts level 2 TLB refills caused by memory write operations from both data and instruction fetch except for those caused by TLB maintenance operations. This event is a subset of the L2D_TLB_REFILL event.
0x005E	L2D_TLB_RD	**Level 2 data TLB access, read** Counts level 2 TLB accesses caused by memory read operations from both data and instruction fetch except for those caused by TLB maintenance operations. This event is a subset of the L2D_TLB event.
0x005F	L2D_TLB_WR	**Level 2 data TLB access, write** Counts level 2 TLB accesses caused by memory write operations from both data and instruction fetch except for those caused by TLB maintenance operations. This event is a subset of the L2D_TLB event.
0x0060	BUS_ACCESS_RD	**Bus access, read** Counts memory read transactions seen on the external bus. Each beat of data is counted individually.
0x0061	BUS_ACCESS_WR	**Bus access, write** Counts memory write transactions seen on the external bus. Each beat of data is counted individually.
0x0066	MEM_ACCESS_RD	**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. The following instructions are not counted: 1) Instruction fetches, 2) Cache maintenance instructions, 3) Translation table walks or prefetches, 4) Memory prefetch operations. This event is a subset of the MEM_ACCESS event but the event only counts memory-read operations.

Event number	Mnemonic	Description
0x0067	MEM_ACCESS_WR	**Data memory access, write** Counts memory accesses issued by the CPU 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. The event also counts atomic load and store 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.
0x0068	UNALIGNED_LD_SPEC	**Unaligned access, read** 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). This event is a subset of the UNALIGNED_LDST_SPEC event.
0x0069	UNALIGNED_ST_SPEC	**Unaligned access, write** 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. This event is a subset of the UNALIGNED_LDST_SPEC event.
0x006A	UNALIGNED_LDST_SPEC	**Unaligned access** 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. This event is the sum of the UNALIGNED_ST_SPEC and UNALIGNED_LD_SPEC events.
0x006C	LDREX_SPEC	**Exclusive operation speculatively executed, Load-Exclusive** Counts Load-Exclusive operations that have been speculatively executed. For example: LDREX, LDX
0x006D	STREX_PASS_SPEC	**Exclusive operation speculatively executed, Store-Exclusive pass** Counts store-exclusive operations that have been speculatively executed and have successfully completed the store operation.
0x006E	STREX_FAIL_SPEC	**Exclusive operation speculatively executed, Store-Exclusive fail** Counts store-exclusive operations that have been speculatively executed and have not successfully completed the store operation.
0x006F	STREX_SPEC	**Exclusive operation speculatively executed, Store-Exclusive** Counts store-exclusive operations that have been speculatively executed. This event is the sum of STREX_PASS_SPEC and STREX_FAIL_SPEC events.
0x0070	LD_SPEC	**Operation speculatively executed, load** Counts speculatively executed load operations including Single Instruction Multiple Data (SIMD) load operations.
0x0071	ST_SPEC	**Operation speculatively executed, store** Counts speculatively executed store operations including Single Instruction Multiple Data (SIMD) store operations.

Event number	Mnemonic	Description
0x0072	LDST_SPEC	**Operation speculatively executed, load or store** Counts load and store operations that have been speculatively executed. Operations that perform a load and a store operation are counted only once in this counter
0x0073	DP_SPEC	**Operation speculatively executed, integer data processing** Counts speculatively executed logical or arithmetic instructions such as MOV/MVN operations.
0x0074	ASE_SPEC	**Operation speculatively executed, Advanced SIMD data processing** 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.
0x0075	VFP_SPEC	**Operation speculatively executed, scalar floating-point** Counts speculatively executed floating point operations. This event does not count operations that move data to or from floating point (vector) registers.
0x0076	PC_WRITE_SPEC	**Operation speculatively executed, Software change of the PC** Counts speculatively executed operations which cause software changes of the PC. Those operations include all taken branch operations.
0x0077	CRYPTO_SPEC	**Operation speculatively executed, Cryptographic instruction** Counts speculatively executed cryptographic operations except for PMULL and VMULL operations.
0x0078	BR_IMMED_SPEC	**Branch speculatively executed, immediate branch** Counts direct branch operations which are speculatively executed.
0x0079	BR_RETURN_SPEC	**Branch speculatively executed, procedure return** Counts procedure return operations (RET, RETAA and RETAB) which are speculatively executed.
0x007A	BR_INDIRECT_SPEC	**Branch speculatively executed, indirect branch** Counts indirect branch operations including procedure returns, which are speculatively executed. This includes operations that force a software change of the PC, other than exception-generating operations and direct branch instructions. Some examples of the instructions counted by this event include BR Xn, RET, etc...
0x007C	ISB_SPEC	**Barrier speculatively executed, ISB** Counts ISB operations that are executed.
0x007D	DSB_SPEC	**Barrier speculatively executed, DSB** Counts DSB operations that are speculatively issued to Load/Store unit in the CPU.
0x007E	DMB_SPEC	**Barrier speculatively executed, DMB** Counts DMB operations that are speculatively issued to the Load/Store unit in the CPU. This event does not count implied barriers from load acquire/store release operations.
0x007F	CSDB_SPEC	**Barrier speculatively executed, CSDB** Counts CDSB operations that are speculatively issued to the Load/Store unit in the CPU. This event does not count implied barriers from load acquire/store release operations.

Event number	Mnemonic	Description
0x0081	EXC_UNDEF	**Exception taken, other synchronous** Counts the number of synchronous exceptions which are taken locally that are due to attempting to execute an instruction that is UNDEFINED. Attempting to execute instruction bit patterns that have not been allocated. Attempting to execute instructions when they are disabled. Attempting to execute instructions at an inappropriate Exception level. Attempting to execute an instruction when the value of PSTATE.IL is 1.
0x0082	EXC_SVC	**Exception taken, Supervisor Call** Counts SVC exceptions taken locally.
0x0083	EXC_PABORT	**Exception taken, Instruction Abort** Counts synchronous exceptions that are taken locally and caused by Instruction Aborts.
0x0084	EXC_DABORT	**Exception taken, Data Abort or SError** Counts exceptions that are taken locally and are caused by data aborts or SErrors. Conditions that could cause those exceptions are attempting to read or write memory where the MMU generates a fault, attempting to read or write memory with a misaligned address, interrupts from the nSEI inputs and internally generated SErrors.
0x0086	EXC_IRQ	**Exception taken, IRQ** Counts IRQ exceptions including the virtual IRQs that are taken locally.
0x0087	EXC_FIQ	**Exception taken, FIQ** Counts FIQ exceptions including the virtual FIQs that are taken locally.
0x0088	EXC_SMC	**Exception taken, Secure Monitor Call** Counts SMC exceptions take to EL3.
0x008A	EXC_HVC	**Exception taken, Hypervisor Call** Counts HVC exceptions taken to EL2.
0x008B	EXC_TRAP_PABORT	**Exception taken, Instruction Abort not Taken locally** Counts exceptions which are traps not taken locally and are caused by Instruction Aborts. For example, attempting to execute an instruction with a misaligned PC.
0x008C	EXC_TRAP_DABORT	**Exception taken, Data Abort or SError not Taken locally** Counts exceptions which are traps not taken locally and are caused by Data Aborts or SError interrupts. Conditions that could cause those exceptions are: 1. Attempting to read or write memory where the MMU generates a fault, 2. Attempting to read or write memory with a misaligned address, 3. Interrupts from the SEI input. 4. internally generated SErrors.
0x008D	EXC_TRAP_OTHER	**Exception taken, other traps not Taken locally** Counts the number of synchronous trap exceptions which are not taken locally and are not SVC, SMC, HVC, data aborts, Instruction Aborts, or interrupts.
0x008E	EXC_TRAP_IRQ	**Exception taken, IRQ not Taken locally** Counts IRQ exceptions including the virtual IRQs that are not taken locally.
0x008F	EXC_TRAP_FIQ	**Exception taken, FIQ not Taken locally** Counts FIQs which are not taken locally but taken from EL0, EL1, or EL2 to EL3 (which would be the normal behavior for FIQs when not executing in EL3).

Event number	Mnemonic	Description
0x0090	RC_LD_SPEC	**Release consistency operation speculatively executed, Load-Acquire** Counts any load acquire operations that are speculatively executed. For example: LDAR, LDARH, LDARB
0x0091	RC_ST_SPEC	**Release consistency operation speculatively executed, Store-Release** Counts any store release operations that are speculatively executed. For example: STLR, STLRH, STLRB
0x00A0	L3D_CACHE_RD	**Level 3 data cache access, read** Counts level 3 cache accesses caused by any memory read 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.
0x4000	SAMPLE_POP	**Statistical Profiling sample population** Counts statistical profiling sample population, the count of all operations that could be sampled but may or may not be chosen for sampling.
0x4001	SAMPLE_FEED	**Statistical Profiling sample taken** Counts statistical profiling samples taken for sampling.
0x4002	SAMPLE_FILTRATE	**Statistical Profiling sample taken and not removed by filtering** Counts statistical profiling samples taken which are not removed by filtering.
0x4003	SAMPLE_COLLISION	**Statistical Profiling sample collided with previous sample** Counts statistical profiling samples that have collided with a previous sample and so therefore not taken.
0x4004	CNT_CYCLES	**Constant frequency cycles** Increments at a constant frequency equal to the rate of increment of the System Counter, CNTPCT_EL0.
0x4005	STALL_BACKEND_MEM	**Memory stall cycles** Counts cycles when the backend is stalled because there is a demand data miss in the last level core cache. Last level cache in this CPU is Level 2, hence this event counts in place of STALL_BACKEND_L2D
0x4006	L1I_CACHE_LMISS	**Level 1 instruction cache long-latency miss** Counts cache line refills into the level 1 instruction cache, that incurred additional latency.
0x4009	L2D_CACHE_LMISS_RD	**Level 2 data cache long-latency read miss** Counts cache line refills into the level 2 unified cache from any memory read operations that incurred additional latency. Counts the same as L2D_CACHE_REFILL_RD in this CPU
0x400A	L2I_CACHE_LMISS	**Level 2 instruction cache long-latency miss** Counts cache line refills into the level 2 unified cache from any instruction read operations that incurred additional latency. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x400B	L3D_CACHE_LMISS_RD	**Level 3 data cache long-latency read miss** Counts any cache line refill into the level 3 cache from memory read operations that incurred additional latency.
0x400C	TRB_WRAP	**Trace buffer current write pointer wrapped** This event is generated each time the current write pointer is wrapped to the base pointer. **Note:** This event is not exported to the trace unit.
0x400D	PMU_OVFS	**PMU overflow, counters accessible to EL1 and EL0** This event is generated each time an event causes a PMEVCTNR<n>_EL1 counter overflow when PMINTENSET_EL1[n] is set to 1, for each implemented PMU counter n in the range 0 <= n < UInt(MDCR_EL2.HPMN), and the Cycle Counter (n = 31). **Note:** This event is exported to the trace unit, but cannot be counted in the PMU.
0x400E	TRB_TRIG	**Trace buffer Trigger Event** This event is generated when a Trace Buffer Extension Trigger Event occurs. **Note:** This event is exported to the trace unit, but cannot be counted in the PMU.
0x400F	PMU_HOVFS	**PMU overflow, counters reserved for use by EL2** This event is generated each time an event causes a PMEVCTNR<n>_EL1 counter overflow when PMINTENSET_EL1[n] is set to 1, for each implemented PMU counter n in the range UInt(MDCR_EL2.HPMN) <= n < UInt(PMCR_EL0.N). This event is not transmitted to a PE Trace Unit while TRCFR_EL2.E2TRE == 0b0. **Note:** This event is exported to the trace unit, but cannot be counted in the PMU.
0x4010	TRCEXTOUT0	**Trace unit external output 0** This event is generated each time an event is signaled by ETE external event 0. **Note:** This event is not exported to the trace unit.
0x4011	TRCEXTOUT1	**Trace unit external output 1** This event is generated each time an event is signaled by ETE external event 1. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x4012	TRCEXTOUT2	**Trace unit external output 2** This event is generated each time an event is signaled by ETE external event 2. **Note:** This event is not exported to the trace unit.
0x4013	TRCEXTOUT3	**Trace unit external output 3** This event is generated each time an event is signaled by ETE external event 3. **Note:** This event is not exported to the trace unit.
0x4018	CTI_TRIGOUT4	**Cross-trigger Interface output trigger 4** This event is generated each time an event is signaled on CTI output trigger 4. **Note:** This event is not exported to the trace unit.
0x4019	CTI_TRIGOUT5	**Cross-trigger Interface output trigger 5** This event is generated each time an event is signaled on CTI output trigger 5. **Note:** This event is not exported to the trace unit.
0x401A	CTI_TRIGOUT6	**Cross-trigger Interface output trigger 6** This event is generated each time an event is signaled on CTI output trigger 6. **Note:** This event is not exported to the trace unit.
0x401B	CTI_TRIGOUT7	**Cross-trigger Interface output trigger 7** This event is generated each time an event is signaled on CTI output trigger 7. **Note:** This event is not exported to the trace unit.
0x4020	LDST_ALIGN_LAT	**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. This event is implemented as the sum of LD_ALIGN_LAT and ST_ALIGN_LAT on this CPU

Event number	Mnemonic	Description
0x4021	LD_ALIGN_LAT	**Load 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.
0x4022	ST_ALIGN_LAT	**Store with additional latency from alignment** Counts the number of memory write access in a cycle that incurred additional latency, due to the alignment of the address and size of data being accessed incurred additional latency.
0x4024	MEM_ACCESS_CHECKED	**Checked data memory access** Counts the number of memory read and write accesses counted by MEM_ACCESS that are tag checked by the Memory Tagging Extension (MTE). This event is implemented as the sum of MEM_ACCESS_CHECKED_RD and MEM_ACCESS_CHECKED_WR
0x4025	MEM_ACCESS_CHECKED_RD	**Checked data memory access, read** Counts the number of memory read accesses in a cycle that are tag checked by the Memory Tagging Extension (MTE).
0x4026	MEM_ACCESS_CHECKED_WR	**Checked data memory access, write** Counts the number of memory write accesses in a cycle that is tag checked by the Memory Tagging Extension (MTE).
0x8004	SIMD_INST_SPEC	**Operation speculatively executed, SIMD** Counts speculatively executed operations that are SIMD or SVE vector operations or Advanced SIMD non-scalar operations.
0x8005	ASE_INST_SPEC	**Operation speculatively executed, Advanced SIMD** The counter counts each Speculatively executed operation due to an A64 Advanced SIMD instruction. It is IMPLEMENTATION DEFINED whether the counter counts operations due to Advanced SIMD scalar instructions.
0x8006	SVE_INST_SPEC	**Operation speculatively executed, SVE, including load and store** The counter counts each Speculatively executed operation due to an SVE instruction. It is IMPLEMENTATION DEFINED whether the counter counts operations due to non-SIMD SVE instructions. Instructions classified as SME instructions and counted by SME_INST_SPEC are not counted by this event.
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.

Event number	Mnemonic	Description
0x8040	INT_SPEC	**Integer operation speculatively executed** Counts speculatively executed integer arithmetic operations. **Note:** This event is not exported to the trace unit.
0x8056	SVE_SPEC	**Operation speculatively executed, SVE** The counter counts each operation counted by INST_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
0x8057	ASE_SVE_SPEC	**Operation speculatively executed, Advanced SIMD or SVE data processing** The counter counts each operation counted by INST_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** Counts speculatively executed predicated SVE operations.
0x8075	SVE_PRED_EMPTY_SPEC	**Operation speculatively executed, SVE predicated with no active predicates** Counts speculatively executed predicated SVE operations with no active predicate elements.
0x8076	SVE_PRED_FULL_SPEC	**Operation speculatively executed, SVE predicated with all active predicates** Counts speculatively executed predicated SVE operations with all predicate elements active.
0x8077	SVE_PRED_PARTIAL_SPEC	**Operation speculatively executed, SVE predicated with partially active predicates** Counts speculatively executed predicated SVE operations with at least one but not all active predicate elements.
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	**SVE predicated operations speculatively executed with no active or partially active predicates** Counts speculatively executed predicated SVE operations with at least one non active predicate elements.

Event number	Mnemonic	Description
0x8080	SVE_LDST_SPEC	**Operation speculatively executed, SVE load, store, or prefetch** The counter counts each Speculatively executed load, store, or prefetch operation counted by any of SVE_LD_SPEC, SVE_PRF_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.
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.
0x8083	SVE_PRF_SPEC	**Operation speculatively executed, SVE prefetch** The counter counts each Speculatively executed prefetch operation due to any of the following A64 instructions: • SVE: PRFB, PRFD, PRFH, or PRFW.
0x8087	PRF_SPEC	**Operation speculatively executed, Prefetch** Counts speculatively executed operations that prefetch memory. For example: Scalar: PRFM, SVE: PRFB, PRFD, PRFH, or PRFW. **Note:** This event is not exported to the trace unit.
0x80BC	SVE_LDFF_SPEC	**Operation speculatively executed, SVE first-fault load** Counts speculatively executed SVE first fault or non-fault load operations.
0x80BD	SVE_LDFF_FAULT_SPEC	**Operation speculatively executed, SVE first-fault load which set FFR bit to 0b0** Counts speculatively executed SVE first fault or non-fault load operations that clear at least one bit in the FFR.
0x80C0	FP_SCALE_OPS_SPEC	**Scalable floating-point element ALU operations speculatively executed** Counts speculatively executed scalable single precision floating point operations.
0x80C1	FP_FIXED_OPS_SPEC	**Non-scalable floating-point element ALU operations speculatively executed** Counts speculatively executed non-scalable single precision floating point operations.
0x80E3	ASE_SVE_INT8_SPEC	**Integer operation speculatively executed, Advanced SIMD or SVE 8-bit** Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type an 8-bit integer.
0x80E7	ASE_SVE_INT16_SPEC	**Integer operation speculatively executed, Advanced SIMD or SVE 16-bit** Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type a 16-bit integer.
0x80EB	ASE_SVE_INT32_SPEC	**Integer operation speculatively executed, Advanced SIMD or SVE 32-bit** Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type a 32-bit integer.
0x80EF	ASE_SVE_INT64_SPEC	**Integer operation speculatively executed, Advanced SIMD or SVE 64-bit** Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type a 64-bit integer.

Event number	Mnemonic	Description
0x8108	BR_IMMED_TAKEN_RETIRED	**Branch instruction architecturally executed, immediate, taken** Counts architecturally executed direct branches that were taken.
0x810C	BR_INDNR_TAKEN_RETIRED	**Branch instruction architecturally executed, indirect excluding procedure return, taken** Counts architecturally executed indirect branches excluding procedure returns that were taken.
0x8110	BR_IMMED_PRED_RETIRED	**Branch instruction architecturally executed, predicted immediate** Counts architecturally executed direct branches that were correctly predicted.
0x8111	BR_IMMED_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted immediate** Counts architecturally executed direct branches that were mispredicted and caused a pipeline flush.
0x8112	BR_IND_PRED_RETIRED	**Branch instruction architecturally executed, predicted indirect** Counts architecturally executed indirect branches including procedure returns that were correctly predicted.
0x8113	BR_IND_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted indirect** Counts architecturally executed indirect branches including procedure returns that were mispredicted and caused a pipeline flush.
0x8114	BR_RETURN_PRED_RETIRED	**Branch instruction architecturally executed, predicted procedure return** Counts architecturally executed procedure returns that were correctly predicted.
0x8115	BR_RETURN_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted procedure return** Counts architecturally executed procedure returns that were mispredicted and caused a pipeline flush.
0x8116	BR_INDNR_PRED_RETIRED	**Branch instruction architecturally executed, predicted indirect excluding procedure return** Counts architecturally executed indirect branches excluding procedure returns that were correctly predicted.
0x8117	BR_INDNR_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted indirect excluding procedure return** Counts architecturally executed indirect branches excluding procedure returns that were mispredicted and caused a pipeline flush.
0x8118	BR_TAKEN_PRED_RETIRED	**Branch instruction architecturally executed, predicted branch, taken** Counts architecturally executed branches that were taken and were correctly predicted.
0x8119	BR_TAKEN_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted branch, taken** Counts architecturally executed branches that were taken and were mispredicted causing a pipeline flush.
0x811A	BR_SKIP_PRED_RETIRED	**Branch instruction architecturally executed, predicted branch, not taken** Counts architecturally executed branches that were not taken and were correctly predicted.
0x811B	BR_SKIP_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted branch, not taken** Counts architecturally executed branches that were not taken and were mispredicted causing a pipeline flush.
0x811C	BR_PRED_RETIRED	**Branch instruction architecturally executed, predicted branch** Counts branch instructions counted by BR_RETIRED which were correctly predicted.
0x811D	BR_IND_RETIRED	**Instruction architecturally executed, indirect branch** Counts architecturally executed indirect branches including procedure returns.

Event number	Mnemonic	Description
0x8120	INST_FETCH_PERCYC	**Event in progress, INST FETCH** Counts number of instruction fetches outstanding per cycle, which will provide an average latency of instruction fetch.
0x8121	MEM_ACCESS_RD_PERCYC	**Event in progress, MEM ACCESS RD** Counts the number of outstanding loads or memory read accesses per cycle.
0x8124	INST_FETCH	**Instruction memory access** Counts Instruction memory accesses that the PE makes.
0x8125	BUS_REQ_RD_PERCYC	**Bus read transactions in progress** Counts memory read transaction requests issued by the CPU to the external bus in progress on a processor cycle.
0x8128	DTLB_WALK_PERCYC	**Event in progress, DTLB WALK** Counts the number of data translation table walks in progress per cycle.
0x8129	ITLB_WALK_PERCYC	**Event in progress, ITLB WALK** Counts the number of instruction translation table walks in progress per cycle.
0x812A	SAMPLE_FEED_BR	**Statistical Profiling sample taken, branch** Counts statistical profiling samples taken which are branches.
0x812B	SAMPLE_FEED_LD	**Statistical Profiling sample taken, load** Counts statistical profiling samples taken which are loads or load atomic operations.
0x812C	SAMPLE_FEED_ST	**Statistical Profiling sample taken, store** Counts statistical profiling samples taken which are stores or store atomic operations.
0x812D	SAMPLE_FEED_OP	**Statistical Profiling sample taken, matching operation type** Counts statistical profiling samples taken which are matching any operation type filters supported.
0x812E	SAMPLE_FEED_EVENT	**Statistical Profiling sample taken, matching events** Counts statistical profiling samples taken which are matching event packet filter constraints.
0x812F	SAMPLE_FEED_LAT	**Statistical Profiling sample taken, exceeding minimum latency** Counts statistical profiling samples taken which are exceeding minimum latency set by operation latency filter constraints.
0x8130	L1D_TLB_RW	**Level 1 data TLB demand access** Counts level 1 data TLB demand accesses caused by memory read or write operations. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.
0x8131	L1I_TLB_RD	**Level 1 instruction TLB demand access** Counts level 1 instruction TLB demand accesses whether the access hits or misses in the TLB.
0x8132	L1D_TLB_PRFM	**Level 1 data TLB software preload** Counts level 1 data TLB accesses generated by software prefetch or preload memory accesses. Load or store instructions can be broken into multiple memory operations. This event does not count TLB maintenance operations.
0x8133	L1I_TLB_PRFM	**Level 1 instruction TLB software preload** Counts level 1 instruction TLB accesses generated by software preload or prefetch instructions. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.

Event number	Mnemonic	Description
0x8134	DTLB_HWUPD	**Data TLB hardware update of translation table** Counts number of memory accesses triggered by a data translation table walk and performing an update of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that this event counts accesses triggered by software preloads, but not accesses triggered by hardware prefetchers.
0x8135	ITLB_HWUPD	**Instruction TLB hardware update of translation table** Counts number of memory accesses triggered by an instruction translation table walk and performing an update of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD.
0x8136	DTLB_STEP	**Data TLB translation table walk, step** Counts number of memory accesses triggered by a demand data translation table walk and performing a read of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that this event counts accesses triggered by software preloads, but not accesses triggered by hardware prefetchers.
0x8137	ITLB_STEP	**Instruction TLB translation table walk, step** Counts number of memory accesses triggered by an instruction translation table walk and performing a read of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD.
0x8138	DTLB_WALK_LARGE	**Data TLB large page translation table walk** Counts number of demand data translation table walks caused by a miss in the L2 TLB and yielding a large page. The set of large pages is defined as all pages with a final size higher than or equal to 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. If DTLB_WALK_BLOCK is implemented, then it is an alias for this event in this family. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations.
0x8139	ITLB_WALK_LARGE	**Instruction TLB large page translation table walk** Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding a large page. The set of large pages is defined as all pages with a final size higher than or equal to 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is equal to ITLB_WALK_BLOCK event. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x813A	DTLB_WALK_SMALL	**Data TLB small page translation table walk** Counts number of data translation table walks caused by a miss in the L2 TLB and yielding a small page. The set of small pages is defined as all pages with a final size lower than 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. If DTLB_WALK_PAGE event is implemented, then it is an alias for this event in this family. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations.

Event number	Mnemonic	Description
0x813B	ITLB_WALK_SMALL	**Instruction TLB small page translation table walk** Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding a small page. The set of small pages is defined as all pages with a final size lower than 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is equal to ITLB_WALK_PAGE event. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x813C	DTLB_WALK_RW	**Data TLB demand access with at least one translation table walk** Counts number of demand data translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x813D	ITLB_WALK_RD	**Instruction TLB demand access with at least one translation table walk** Counts number of demand instruction translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x813E	DTLB_WALK_PRFM	**Data TLB software preload access with at least one translation table walk** Counts number of software prefetches or preloads generated data translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x813F	ITLB_WALK_PRFM	**Instruction TLB software preload access with at least one translation table walk** Counts number of software prefetches or preloads generated instruction translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x8140	L1D_CACHE_RW	**Level 1 data cache demand access** Counts level 1 data demand cache accesses from any load or store operation. Near atomic operations that resolve in the CPUs caches counts as both a write access and read access. This event is implemented as L1D_CACHE_RD + L1D_CACHE_WR
0x8141	L1I_CACHE_RD	**Level 1 instruction cache demand fetch** Counts demand instruction fetches which access the level 1 instruction cache.
0x8142	L1D_CACHE_PRFM	**Level 1 data cache software preload** Counts level 1 data cache accesses from software preload or prefetch instructions.
0x8143	L1I_CACHE_PRFM	**Level 1 instruction cache software preload** Counts instruction fetches generated by software preload or prefetch instructions which access the level 1 instruction cache.

Event number	Mnemonic	Description
0x8144	L1D_CACHE_MISS	**Level 1 data cache demand access miss** Counts cache line misses in the level 1 data cache.
0x8145	L1I_CACHE_HWPRF	**Level 1 instruction cache hardware prefetch** Counts instruction fetches which access the level 1 instruction cache generated by the hardware prefetcher.
0x8146	L1D_CACHE_REFILL_PRFM	**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.
0x8147	L1I_CACHE_REFILL_PRFM	**Level 1 instruction cache refill, software preload** Counts cache line refills in the level 1 instruction cache caused by a missed instruction fetch generated by software preload or prefetch instructions. Instruction fetches may include accessing multiple instructions, but the single cache line allocation is counted once.
0x8148	L2D_CACHE_RW	**Level 2 data cache demand access** Counts level 2 cache demand accesses from any load/store operations. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This event is the sum of the L2D_CACHE_RD and L2D_CACHE_WR events.
0x8149	L2I_CACHE_RD	**Level 2 instruction cache demand fetch** Counts level 2 cache accesses that are due to a demand instruction cache access. Counts same as L1I_CACHE_REFILL_RD on this CPU.
0x814A	L2D_CACHE_PRFM	**Level 2 data cache software preload** Counts level 2 data cache accesses generated by software preload or prefetch instructions.
0x814B	L2I_CACHE_PRFM	**Level 2 instruction cache software preload** Counts level 2 instruction cache accesses generated by software preload or prefetch instructions. **Note:** This event is not exported to the trace unit.
0x814C	L2D_CACHE_MISS	**Level 2 data cache demand access miss** Counts cache line misses in the level 2 cache. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the level 1 data cache or translation resolutions due to accesses. Counts same as L2D_CACHE_REFILL in this CPU
0x814E	L2D_CACHE_REFILL_PRFM	**Level 2 data cache refill, software preload** Counts refills due to accesses generated as a result of software preload or prefetch instructions as counted by L2D_CACHE_PRFM.
0x814F	L2I_CACHE_REFILL_PRFM	**Level 2 instruction cache refill, software preload** Counts level 2 instruction cache refills generated by software preload or prefetch instructions.

Event number	Mnemonic	Description
0x8150	L3D_CACHE_RW	**Level 3 data cache demand access** 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.
0x8151	L3D_CACHE_PRFM	**Level 3 data cache software preload** Counts level 3 cache accesses generated by hardware or software prefetches.
0x8152	L3D_CACHE_MISS	**Level 3 data cache demand access miss** Counts level 3 cache accesses that missed in the level 3 cache.
0x8153	L3D_CACHE_REFILL_PRFM	**Level 3 data cache refill, software preload** Counts cacheable reads generated by hardware or software prefetches that receive data from outside the L3 cache.
0x8154	L1D_CACHE_HWPRF	**Level 1 data cache hardware prefetch** Counts level 1 data cache accesses from any load/store operations generated by the hardware prefetcher.
0x8155	L2D_CACHE_HWPRF	**Level 2 data cache hardware prefetch** Counts level 2 data cache accesses generated by L2D hardware prefetchers.
0x8158	STALL_FRONTEND_MEMBOUND	**Frontend stall cycles, memory bound** Counts cycles when the frontend could not send any micro-operations to the rename stage due to resource constraints in the memory resources.
0x8159	STALL_FRONTEND_L1I	**Frontend stall cycles, level 1 instruction cache** Counts cycles when the frontend is stalled because there is an instruction fetch request pending in the level 1 instruction cache.
0x815B	STALL_FRONTEND_MEM	**Frontend stall cycles, last level PE cache or memory** Counts cycles when the frontend is stalled because there is an instruction fetch request pending in the last level core cache. Last level cache in this CPU is Level 2, hence this event counts in place of STALL_FRONTEND_L2I
0x815C	STALL_FRONTEND_TLB	**Frontend stall cycles, TLB** Counts when the frontend is stalled on any TLB misses being handled. This event also counts the TLB accesses made by hardware prefetches.
0x8160	STALL_FRONTEND_CPUBOUND	**Frontend stall cycles, processor bound** Counts cycles when the frontend could not send any micro-operations to the rename stage due to resource constraints in the CPU resources excluding memory resources.
0x8161	STALL_FRONTEND_FLOW	**Frontend stall cycles, flow control** Counts cycles when the frontend could not send any micro-operations to the rename stage due to resource constraints in the branch prediction unit.
0x8162	STALL_FRONTEND_FLUSH	**Frontend stall cycles, flush recovery** Counts cycles when the frontend could not send any micro-operations to the rename stage as the frontend is recovering from a machine flush or resteer. Example scenarios that cause a flush include branch mispredictions, taken exceptions, micro-architectural flush etc.
0x8164	STALL_BACKEND_MEMBOUND	**Backend stall cycles, memory bound** Counts cycles when the backend could not accept any micro-operations due to resource constraints in the memory resources.

Event number	Mnemonic	Description
0x8165	STALL_BACKEND_L1D	**Backend stall cycles, level 1 data cache** Counts cycles when the backend is stalled because there is a demand data miss in the level 1 data cache.
0x8167	STALL_BACKEND_TLB	**Backend stall cycles, TLB** Counts cycles when the backend is stalled on any demand TLB misses being handled.
0x8168	STALL_BACKEND_ST	**Backend stall cycles, store** Counts cycles when the backend is stalled and there is a store that has not reached the pre-commit stage.
0x816A	STALL_BACKEND_CPUBOUND	**Backend stall cycles, processor bound** Counts cycles when the backend could not accept any micro-operations due to any resource constraints in the CPU excluding memory resources.
0x816B	STALL_BACKEND_BUSY	**Backend stall cycles, backend busy** Counts cycles when the backend could not accept any micro-operations because the issue queues are full to take any operations for execution.
0x816D	STALL_BACKEND_RENAME	**Backend stall cycles, rename full** Counts cycles when backend is stalled even when operations are available from the frontend but at least one is not ready to be sent to the backend because no rename register is available.
0x8170	CAS_NEAR_FAIL	**Atomic memory Operation speculatively executed, Compare and Swap fail** Counts compare and swap instructions that executed locally to the PE and did not update the location accessed.
0x8171	CAS_NEAR_PASS	**Atomic memory Operation speculatively executed, Compare and Swap pass** Counts compare and swap instructions that executed locally to the PE and updated the location accessed.
0x8172	CAS_NEAR_SPEC	**Atomic memory Operation speculatively executed, Compare and Swap near** Counts compare and swap instructions that executed locally to the PE.
0x8173	CAS_FAR_SPEC	**Atomic memory Operation speculatively executed, Compare and Swap far** Counts compare and swap instructions that did not execute locally to the PE.
0x8174	CAS_SPEC	**Atomic memory Operation speculatively executed, Compare and Swap** Counts the total compare and swap instructions that were executed.
0x8175	LSE_LD_SPEC	**Atomic memory Operation speculatively executed, load** Counts the total atomic memory instructions that return a value that were speculatively executed.
0x8176	LSE_ST_SPEC	**Atomic memory Operation speculatively executed, store** Counts the total atomic memory instructions that do not return a value that were speculatively executed.
0x8177	LSE_LDST_SPEC	**Atomic memory Operation speculatively executed, load or store** Counts the total atomic memory instructions that were speculatively executed.
0x8179	BRNL_INDNR_TAKEN_RETIRED	**Branch instruction architecturally executed, indirect branch without link excluding procedure return, taken** Counts architecturally executed indirect branch without link instructions, excluding return instructions, that were taken.
0x817A	BL_TAKEN_RETIRED	**Branch instruction architecturally executed, branch with link, taken** Counts architecturally executed branch with link instructions that were taken.

Event number	Mnemonic	Description
0x817B	BRNL_TAKEN_RETIRED	**Branch instruction architecturally executed, branch without link, taken** Counts architecturally executed branch without link instructions that were taken.
0x817C	BL_IND_TAKEN_RETIRED	**Branch instruction architecturally executed, indirect branch with link, taken** Counts architecturally executed indirect branch with link instructions that were taken.
0x817D	BRNL_IND_TAKEN_RETIRED	**Branch instruction architecturally executed, indirect branch without link, taken** Counts architecturally executed indirect branch without link instructions that were taken.
0x817E	BL_IMMED_TAKEN_RETIRED	**Branch instruction architecturally executed, direct branch with link, taken** Counts architecturally executed direct branch with link instructions that were taken.
0x817F	BRNL_IMMED_TAKEN_RETIRED	**Branch instruction architecturally executed, direct branch without link, taken** Counts architecturally executed direct branch without link instructions that were taken.
0x8180	BR_UNCOND_RETIRED	**Branch instruction architecturally executed, unconditional branch** Counts architecturally executed unconditional branch instructions.
0x8181	BR_COND_RETIRED	**Branch instruction architecturally executed, conditional branch** Counts architecturally executed conditional branch instructions.
0x8188	DTLB_WALK_BLOCK	**Data TLB block translation table walk** Counts number of demand data translation table walks caused by a miss in the L2 TLB and yielding a block descriptor at level 1 or level 2 of the translation. This means that the walk yielded a final size higher than or equal to 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is an alias to DTLB_WALK_LARGE event. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations.
0x8189	ITLB_WALK_BLOCK	**Instruction TLB block translation table walk** Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding a block descriptor at level 1 or level 2 of the translation. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is equal to ITLB_WALK_LARGE event. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x818A	DTLB_WALK_PAGE	**Data TLB page translation table walk** Counts number of demand data translation table walks caused by a miss in the L2 TLB and yielding a page descriptor at level 3 of the translation. This means that the walk yielded a final size lower than 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is an alias to DTLB_WALK_SMALL event. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations.

Event number	Mnemonic	Description
0x818B	ITLB_WALK_PAGE	**Instruction TLB page translation table walk** Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding a page descriptor at level 3 of the translation. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is equal to ITLB_WALK_SMALL event. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x818D	BUS_REQ_RD	**Bus request, read** Counts memory read transaction requests issued by the CPU to the external bus.
0x818E	BUS_REQ_WR	**Bus request, write** Counts memory write transaction requests issued by the CPU to the external bus.
0x818F	BUS_REQ	**Bus request** Counts memory transaction requests issued by the CPU to the external bus.
0x8190	ISNP_HIT_RD	**Snoop hit, demand instruction fetch** Counts each instruction access that is satisfied by a snoop request that hits in a cache outside of the cache hierarchy of this PE **Note:** This event is not exported to the trace unit.
0x81A4	DSNP_HIT_HWPRF	**Snoop hit, hardware data prefetch** Counts each data hardware prefetch that is satisfied by a snoop request that hits in a cache outside of the cache hierarchy of this PE. **Note:** This event is not exported to the trace unit.
0x81B4	DSNP_HIT	**Snoop hit, data** Counts each data access that is satisfied by a snoop request that hits in a cache outside of the cache hierarchy of this PE.
0x81BC	L1D_CACHE_REFILL_HWPRF	**Level 1 data cache refill, hardware prefetch** Counts level 1 data cache refills where the cache line is requested by a hardware prefetcher.
0x81BD	L2D_CACHE_REFILL_HWPRF	**Level 2 data cache refill, hardware prefetch** Counts level 2 data cache refills where the cache line is requested by a hardware prefetcher.
0x81C0	L1I_CACHE_HIT_RD	**Level 1 instruction cache demand fetch hit** Counts demand instruction fetches that access the level 1 instruction cache and hit in the L1 instruction cache.
0x81C1	L2I_CACHE_HIT_RD	**Level 2 instruction cache demand fetch hit** Counts level 2 cache hits that are due to a demand instruction cache access.
0x81D0	L1I_CACHE_HIT_RD_FPRFM	**Level 1 instruction cache demand fetch first hit, fetched by software preload** Counts demand instruction fetches that access the level 1 instruction cache that hit in the L1 instruction cache and the line was requested by a software prefetch.

Event number	Mnemonic	Description
0x81E0	L1I_CACHE_HIT_RD_FHWPRF	**Level 1 instruction cache demand fetch first hit, fetched by hardware prefetcher** Counts demand instruction fetches generated by hardware prefetch that access the level 1 instruction cache and hit in the L1 instruction cache.
0x81EC	L1D_CACHE_HIT_RW_FHWPRF	**Level 1 data cache demand access first hit, fetched by hardware prefetcher** Counts first level 1 data demand cache hit from any load or store operation where the cache line was fetched by a hardware prefetcher.
0x81ED	L2D_CACHE_HIT_RW_FHWPRF	**Level 2 data cache demand access first hit, fetched by hardware prefetcher** Counts first level 2 data demand cache hit from any load or store operation where the cache line was fetched by a hardware prefetcher. For the L2D_CACHE_HIT events, an L2 cache hit is only counted if the operation is satisfied internally by L2 (i.e. no bus request, no snoop to lower level). **Note:** This event is not exported to the trace unit.
0x8200	L1I_CACHE_HIT	**Level 1 instruction cache hit** Counts instruction fetches that access the level 1 instruction cache and hit in the level 1 instruction cache. Instruction cache accesses caused by cache maintenance operations are not counted. **Note:** This event is not exported to the trace unit.
0x8206	L3D_CACHE_HIT	**Level 3 data cache hit** Counts each access counted by L3D_CACHE that hits in the Level 3 cache. 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. **Note:** This event is not exported to the trace unit.
0x8207	LL_CACHE_HIT	**Last level cache hit** Counts each access counted by LL_CACHE that hits in the Last level cache. This event counts transactions for external last level cache when the system register CPUECTLR.EXTLLC bit is set. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x8208	L1I_CACHE_HIT_PRFM	**Level 1 instruction cache software preload hit** Counts instruction fetches generated by software preload or prefetch instructions that access the level 1 instruction cache and hit in the level 1 instruction cache. **Note:** This event is not exported to the trace unit.
0x8209	L2I_CACHE_HIT_PRFM	**Level 2 instruction cache software preload hit** Counts level 2 instruction cache hits generated by software preload or prefetch instructions.
0x8240	L1I_LFB_HIT_RD	**Level 1 instruction cache demand fetch line-fill buffer hit** Counts demand instruction fetches that access the level 1 instruction cache and hit in a line that is in the process of being loaded into the level 1 instruction cache. **Note:** This event is not exported to the trace unit.
0x8244	L1D_LFB_HIT_RD	**Level 1 data cache demand line-fill buffer hit, read** Counts load data accesses that access the level 1 data cache and hit in a line that is in the process of being loaded into the level 1 data cache.
0x8248	L1D_LFB_HIT_WR	**Level 1 data cache demand access line-fill buffer hit, write** Counts store data accesses that access the level 1 data cache and hit in a line that is in the process of being loaded into the level 1 data cache.
0x824C	L1D_LFB_HIT_RW	**Level 1 data cache demand access line-fill buffer hit** Counts load or store data accesses that access the level 1 data cache and hit in a line that is in the process of being loaded into the level 1 data cache.
0x824D	L2D_LFB_HIT_RW	**Level 2 data cache demand access line-fill buffer hit** Counts load or store data accesses that access the level 2 data cache and hit in a line that is in the process of being loaded into the level 2 data cache. **Note:** This event is not exported to the trace unit.
0x8250	L1I_LFB_HIT_RD_FPRFM	**Level 1 instruction cache demand fetch line-fill buffer first hit, recently fetched by software preload** Counts demand instruction fetches generated by software prefetch instructions that access the level 1 instruction cache and hit in a line that is in the process of being loaded into the level 1 instruction cache. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x8260	L1I_LFB_HIT_RD_FHWPRF	**Level 1 instruction cache demand fetch line-fill buffer first hit, recently fetched by hardware prefetcher** Counts demand instruction fetches generated by hardware prefetch that access the level 1 instruction cache and hit in a line that is in the process of being loaded into the level 1 instruction cache. **Note:** This event is not exported to the trace unit.
0x826C	L1D_LFB_HIT_RW_FHWPRF	**Level 1 data cache demand access line-fill buffer first hit, recently fetched by hardware prefetcher** Counts first load or store data accesses that access the level 1 data cache and hit in a line that is in the process of being loaded into the level 1 data cache.
0x826D	L2D_LFB_HIT_RW_FHWPRF	**Level 2 data cache demand access line-fill buffer first hit, recently fetched by hardware prefetcher** Counts first load or store data access that accesses the level 2 cache and hit in a line that is in the process of being loaded into the level 2 cache.
0x8284	L1D_CACHE_PRF	**Level 1 data cache, preload or prefetch hit** Counts level 1 data cache accesses from software preload or prefetch instructions or hardware prefetcher.
0x8285	L2D_CACHE_PRF	**Level 2 data cache, preload or prefetch hit** Counts level 2 data cache accesses from software preload or prefetch instructions or hardware prefetcher. **Note:** This event is not exported to the trace unit.
0x835D	SE_SPEC	**Operation speculatively executed, Advanced SIMD, SVE or SME 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. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
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. **Note:** This event is not exported to the trace unit.
0x835F	SE_INST_SPEC	**Operation speculatively executed, Advanced SIMD, SVE, SME** The counter counts each speculatively executed operation counted by INST_SPEC that is classified as an Advanced SIMD, scalable vector extension, or scalable matrix extension operation. See ASE_SVE_INST_SPEC and SME_INST_SPEC for these classifications **Note:** This event is not exported to the trace unit.
0x8380	ZA_ACTIVE	**PSTATE.ZA active cycles** The counter counts each cycle counted by CPU_CYCLES when PSTATE.ZA was enabled. **Note:** This event is not exported to the trace unit.

17.2 Implementation-defined PMU events

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

Implementation-defined PMU events

The C1-Ultra core Performance Monitoring Unit (PMU) collects IMPLEMENTATION DEFINED events. The following table shows the IMPLEMENTATION DEFINED performance monitors events. These events are not exported to the trace unit. The table also shows the bit position of each event on the event bus. Event numbers that are not listed are reserved.

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

Table 17-2: impdef PMU events

Event number	Mnemonic	Description
0x010B	IMP_L2_CACHE_PREFETCH_LATE	**Late prefetch requests to L2 cache** Counts level 2 cache demand accesses for which prefetch requests were late. This event counts any lookups in the L2 cache that occur after a prefetch request was generated by the prefetcher, but before the prefetched cache line is allocated into the cache. It indicates that the prefetch was useful but not on time. **Note:** This event is not exported to the trace unit.
0x010F	IMP_MMU_L2_TLB_REFILL_PREFETCH_READ	**Level 2 data TLB refill, read, prefetch** Counts the number of L2 Unified TLB refills that occur because of a prefetched read, including hardware and software prefetched reads. It counts both data and instructions. **Note:** This event is not exported to the trace unit.
0x0120	IMP_CT_FLUSH	**Pipeline flushes in the commit unit** Counts the number of pipeline flushes in the commit unit. These flushes refer to architectural and microarchitectural flushes that are resolved when an instruction is no longer speculative. It also includes speculative branch redirects for mispredicted branches. These redirects are not necessarily full pipeline flushes, and they may occur before the branch is the oldest instruction in flight. **Note:** This event is not exported to the trace unit.
0x0121	IMP_CT_FLUSH_MEM_HAZARD	**Non-speculative pipeline flushes in the commit unit due to memory hazards** Counts the number of non-speculative pipeline flushes in the commit unit caused by memory hazards, including Read-After-Write, Write-After- Read, and Write-After-Write hazards. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x0122	IMP_CT_FLUSH_BAD_BRANCH	**Pipeline flushes in the commit unit due to non-branch instructions predicted as branch** Counts the number of pipeline flushes in the commit unit caused by a non-branch instruction predicted as a branch. This type of flush occurs when the branch predictor incorrectly identifies a regular instruction as a branch, which causes speculative execution to follow an incorrect control flow path. When the CPU later detects this misprediction, it must flush the pipeline and restart execution from the correct instruction.Counts flushes due to non-branch instructions predicted as a branch. **Note:** This event is not exported to the trace unit.
0x0124	IMP_CT_FLUSH_ISB	**Pipeline flushes in the commit unit due to Instruction Synchronization Barrier or similar** Counts the number of pipeline flushes in the commit unit caused by an Instruction Synchronization Barrier (ISB) or an instruction that causes similar side effects. An ISB is a barrier instruction used to ensure that all previous instructions have completed before any subsequent instructions are fetched and executed. It is commonly used when switching execution contexts, modifying system registers, or updating the instruction-related state, for example, after self-modifying code. ISBs are necessary for correctness. However, they introduce pipeline stalls, because they force instruction fetch and decode stages to restart. **Note:** This event is not exported to the trace unit.
0x0125	IMP_CT_FLUSH_OTHER	**Pipeline flushes in the commit unit due to other hazards** Counts the number of pipeline flushes in the commit unit that are caused by rare instruction sequences or microarchitecture states. **Note:** This event is not exported to the trace unit.
0x0127	IMP_LS_RAR_HAZARD	**Generated load store hazards due to a Read-After-Read ordering hazard** Counts any load store detected hazards that are generated due to a Read-After-Read (RAR) ordering hazard. This hazard occurs when a load operation is incorrectly speculated or executed out-of-order relative to an earlier load. It leads to potential data inconsistencies. To maintain memory ordering correctness, the CPU invalidates the speculatively executed load instructions and reissues them in the correct order. It flushes the pipeline and execution stalls. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x0128	IMP_LS_RAW_HAZARD	**Generated load store hazards due to a Read-After-Write ordering hazard** Counts any load store detected hazards that are generated due to a Read-After-Write (RAW) ordering hazard. This hazard occurs when a younger load instruction executes before an older store to the same memory location. It leads to incorrect data being read. The CPU detects such violations and triggers a load store flush. It clears the pipeline and re-executes affected instructions to ensure a correct execution order. **Note:** This event is not exported to the trace unit.
0x0158	IMP_STALL_BACKEND_RENAME_FRF	**Backend rename stall due to no available flag registers** Counts the number of backend rename stall cycles due to the flag registers being full (FRF), that is, no flag registers are available. This event counts when an operation is available to be sent to the backend but cannot be sent because all physical flag registers are in use. Flag registers store condition codes such as zero, carry, overflow, and negative flags. These registers are used for conditional execution, comparisons, and arithmetic operations. **Note:** This event is not exported to the trace unit.
0x0159	IMP_STALL_BACKEND_RENAME_GRF	**Backend rename stall due to no available general purpose registers** Counts the number of backend rename stall cycles due to the general purpose registers being full (GRF), that is, no general purpose registers are available. This event counts when an operation is available to be sent to the backend but cannot be sent because all physical general purpose registers are in use.i These registers store temporary data operands and computational results. Their availability is crucial to sustain instruction throughput. **Note:** This event is not exported to the trace unit.
0x015A	IMP_STALL_BACKEND_RENAME_VRF	**Backend rename stall due to no available vector registers** Counts the number of backend rename stall cycles due to the vector registers being full (VRF), that is, no vector registers are available. This event counts when an operation is available to be sent to the backend but cannot be sent because all physical vector registers are in use. Vector registers store Single Instruction, Multiple DATA (SIMD) and Scalable Vector Extension (SVE) operations. Their availability is critical to accelerate parallel computations. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x015C	IMP_STALL_BACKEND_IQ_SX	**Backend dispatch stall due to no room for operations in simple integer issue queues** Counts the number of dispatch stall cycles due to simple integer issue queues (SX IQ) which cannot take any operations for execution. This event counts when the oldest operation is waiting to issue to the SX IQ but it is full. **Note:** This event is not exported to the trace unit.
0x015D	IMP_STALL_BACKEND_IQ_MX	**Backend dispatch stall due to no room for operations in complex integer issue queues** Counts the number of dispatch stall cycles due to complex integer issue queues (MX IQ) which cannot take any operations for execution. This event counts when the oldest operation is waiting to issue to the MX IQ but it is full. **Note:** This event is not exported to the trace unit.
0x015E	IMP_STALL_BACKEND_IQ_LS	**Backend dispatch stall due to no room for operations in load store issue queues** Counts the number of dispatch stall cycles due to load store issue queues (LS IQ) which cannot take any operations for execution. This event counts when the oldest operation is waiting to issue to the LS IQ but it is full. **Note:** This event is not exported to the trace unit.
0x015F	IMP_STALL_BACKEND_IQ_VX	**Backend dispatch stall due to no room for operations in vector issue queues** Counts the number of dispatch stall cycles due to vector issue queues (VX IQ) which cannot take any operations for execution. This event counts when the oldest operation is waiting to issue to the VX IQ but it is full. **Note:** This event is not exported to the trace unit.
0x0160	IMP_STALL_BACKEND_MCQ	**Backend dispatch stage stall, due to full commit queue** Counts cycles where the dispatch stage is stalled because the commit queue (MCQ) is full and cannot take any operations for execution. The commit queue is responsible for managing the final stage of instruction execution, where instructions are retired in program order after completing execution. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x0179	IMP_L2_CACHE_PREFETCH_USEFUL	**L2 cache lookups for lines allocated by prefetch** Counts level 2 cache lookups for load store instructions including store, software prefetch and L1 data prefetch that completed with cache hit on a prefetched line in L2 cache. This event counts L2 cache hits due to a data access on a line that was prefetched by hardware. It does not count if the cache line was not accessed by the CPU but instead was accessed by a snoop request from another CPU. **Note:** This event is not exported to the trace unit.
0x0198	IMP_L2_CHI_RX_CBUSY_0	**Received RXDAT or RXRSP responses, with CBusy 0** Counts the number of RXDAT or RXRSP responses received with a CBusy value of 0. Note that if an RXDAT flit and RXRSP flit, both with a CBusy value of 0 arrive at the same time this event increments only once. **Note:** This event is not exported to the trace unit.
0x0199	IMP_L2_CHI_RX_CBUSY_1	**Received RXDAT or RXRSP responses, with CBusy 1** Counts the number of RXDAT or RXRSP responses received with a CBusy value of 1. Note that if an RXDAT flit and RXRSP flit, both with a CBusy value of 1 arrive at the same time this event increments only once. **Note:** This event is not exported to the trace unit.
0x019A	IMP_L2_CHI_RX_CBUSY_2	**Received RXDAT or RXRSP responses, with CBusy 2** Counts the number of RXDAT or RXRSP responses received with a CBusy value of 2. Note that if an RXDAT flit and RXRSP flit, both with a CBusy value of 2 arrive at the same time this event increments only once. **Note:** This event is not exported to the trace unit.
0x019B	IMP_L2_CHI_RX_CBUSY_3	**Received RXDAT or RXRSP responses, with CBusy 3** Counts the number of RXDAT or RXRSP responses received with a CBusy value of 3. Note that if an RXDAT flit and RXRSP flit, both with a CBusy value of 3 arrive at the same time this event increments only once. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x019C	IMP_L2_CHI_RX_CBUSY_MT	**Received RXDAT or RXRSP responses, with CBusy multi-threaded set** Counts the number of RXDAT or RXRSP responses received with CBusy multi-threaded set. Note that if an RXDAT flit and RXRSP flit, both with a CBusy multi-threaded set arrive at the same time this event increments only once. **Note:** This event is not exported to the trace unit.
0x01B8	IMP_L2D_CACHE_L1HWPRF	**L2D cache access due to L1 hardware prefetch** Counts level 2 cache accesses due to level 1 data cache hardware prefetcher. **Note:** This event is not exported to the trace unit.
0x01B9	IMP_L2D_CACHE_REFILL_L1HWPRF	**L2D cache refill due to L1 hardware prefetch** Counts level 2 cache refills where the cache line is requested by a level 1 data cache hardware prefetcher. **Note:** This event is not exported to the trace unit.
0x0225	IMP_WFX_CLOCK_CYCLES	**Cycles in WFx awake handling snoop** Counts cycles in WFx wait state awake handling external snoop traffic. **Note:** This event is not exported to the trace unit.
0x0399	IMP_STALL_BACKEND_RENAME_PDRF	**Backend rename stall due to no available predicate registers** Counts the number of backend rename stall cycles due to the predicate registers being full (PDRF), that is, no predicate registers are available. **Note:** This event is not exported to the trace unit.
0x1338	IMP_STALL_BACKEND_CPUBOUND_OTHER	**Stall backend cpubound other than renames or issue queue** Counts cycles when backend is stalled even when operations are available from the frontend but at least one is not ready to be sent to the backend because a cpu resource other than renames or issue queues. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x3000	IMP_OP_BRU_ISSUE	**Branch operation issued** This event counts each resolution from the branch execution pipelines. **Note:** This event is not exported to the trace unit.
0x3001	IMP_OP_DPU_ISSUE	**Integer operation issued** This event counts each resolution from the integer execution pipelines. **Note:** This event is not exported to the trace unit.
0x3002	IMP_OP_VPU_ISSUE	**Vector/float operation issued** This event counts each resolution from the vector execution pipelines. **Note:** This event is not exported to the trace unit.
0x3003	IMP_OP_LSU_ISSUE	**Load/store operation issued** This event counts each resolution from the load store execution pipelines. **Note:** This event is not exported to the trace unit.
0x3004	IMP_OP_STD_ISSUE	**Store data operation issued** This event counts each resolution for store data operations. **Note:** This event is not exported to the trace unit.
0x3005	IMP_STALL_FRONTEND_SPEC_THROT	**Stall frontend cycles due to power throttling linked to low confidence branches** Counts cycles when the frontend did not send any micro-operations to the rename stage as the frontend is actively throttle throughput based on speculation around low confidence branches **Note:** This event is not exported to the trace unit.
0x3006	IMP_STALL_FRONTEND_FLUSH_CLEAR	**Stall frontend flush cycles due to architectural or microarchitectural flushes** Counts cycles when the frontend could not send any micro-operations to the rename stage as the frontend is recovering from a machine flush due to taken exceptions or other micro-architectural flushes not related to branch mispredictions. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x3007	IMP_STALL_FRONTEND_FLUSH_RESTEER	**Stall frontend flush cycles due to flush for branch mispredictions** Counts cycles when the frontend could not send any micro-operations to the rename stage as the frontend is recovering from a resteer due to branch mispredictions. **Note:** This event is not exported to the trace unit.
0x3008	IMP_DRAM_ACCESS	**Count of loads that were completed at DRAM** Counts access where the data was sourced from the DRAM. **Note:** This event is not exported to the trace unit.
0x3009	IMP_STALL_BACKEND_SPEC_THROT	**Stall backend cycles due to power throttling linked to low confidence branches** Counts cycles when the backend did not accept any micro-operations as the backend is actively throttle throughput based on speculation around low confidence branches **Note:** This event is not exported to the trace unit.
0x3200	STALL_BACKEND_BUSY_CME	**Backend stall cycles, SME2 unit busy** The counter counts each PE cycle counted by STALL_BACKEND_CPUBOUND when the PEs backend was stalled due SME instructions not able to progress, typically: • When waiting for SME2 unit arbitration • Because of SME2 unit backpressure • Because instructions cannot be sent to the SME2 unit due to dependencies, commit, etc. **Note:** This event is not exported to the trace unit.
0x3201	STALL_BACKEND_BUSY_CMEBOUND	**Backend stall cycles, SME2 unit backpressure** The counter counts each CPU cycle counted by STALL_BACKEND_BUSY_CME when the SME2 unit causes backpressure and does not accept instructions. Notes: • This event counts independently of oldest instruction being ready to be sent to the SME2 unit • This event does not count when the CPU is waiting for arbitration, and STALL_BACKEND_BUSY_CME_ARB is counting. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x3202	STALL_BACKEND_BUSY_CME_ARB	**Backend stall cycles, SME2 unit arbitration** The counter counts each CPU cycle counted by STALL_BACKEND_BUSY_CME when oldest SME instruction cannot be sent because it is waiting for arbitration. **Note:** This event is not exported to the trace unit.
0x3203	STALL_BACKEND_BUSY_CME_CPUBOUND	**Backend stall cycles, SME2 unit stalled by CPU** The counter counts each PE cycle counted by STALL_BACKEND_BUSY_CME when not counted by STALL_BACKEND_BUSY_CME_BOUND or STALL_BACKEND_BUSY_CME_ARB. This can typically be due to: • Instruction waiting for commit point being reached • A register dependency prevents the SSVE oldest instruction to be sent • A control packet needs to be sent **Note:** This event is not exported to the trace unit.
0x320C	STALL_BACKEND_MEM_CME_LSRT_FULL	**Backend stall cycles, SME2 unit stalled on LSRT full** The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one Streaming SVE instruction is waiting for an entry being freed to be allocated into the LSRT. **Note:** This event is not exported to the trace unit.
0x320D	STALL_BACKEND_MEM_CME_BARRIER	**Backend stall cycles, barrier stalled by SME2 unit** The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when a barrier instruction is executed and waits for completions from SME load/store transactions. This includes effect due to store-release or load-acquire semantic. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x320E	STALL_BACKEND_MEM_CME_HZ_ON_CPU	**Backend stall cycles, SME2 unit stalled by CPU memory hazard** The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one Streaming SVE load/store instruction is waiting for resolution from an address hazard. This could be used to detect cases for which the CPU and SME2 unit are making overlapping accesses, that is, both are accessing the same location, and the SME2 unit cannot accept the operation to preserve the ordering of memory effects for the location required by the architecture. **Note:** This event is not exported to the trace unit.
0x320F	STALL_BACKEND_MEM_CPU_HZ_ON_CME	**Backend stall cycles, CPU stalled by SME2 unit memory hazard** The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one CPU load/store instruction is waiting for resolution from an address hazard. This could be used to detect cases for which the CPU and SME2 unit are making overlapping accesses, that is, both are accessing the same location, and the CPU cannot execute the operation to preserve the ordering of memory effects for the location required by the architecture. **Note:** This event is not exported to the trace unit.
0x3210	STALL_BACKEND_MEM_CME	**Backend stall cycles, SME2 unit stalled on memory hazard or LSRT full** The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one Streaming SVE instruction is waiting for an entry being freed to be allocated into the LSRT or an address hazard to be resolved, or at least one CPU load/store instruction is waiting for resolution from an address hazard caused by Streaming SVE instruction. **Note:** This event is not exported to the trace unit.
0x3212	SM_ACTIVE_CYCLES	**Cycles PSTATE.SM enabled** The counter counts each cycle counted by CPU_CYCLES when PSTATE.SM was enabled. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x3213	CYCLES_CME_ALLOC	**Cycles SME2 unit allocated** The counter counts each PE cycle counted by CPU_CYCLES where the CPU had an granted arbitration to the SME2 unit, such that the SME2 and Streaming SVE state of the CPU is held in that SME2 unit. It does not count cycles during which a CPU has requested arbitration and is waiting for acknowledgement. **Note:** This event is not exported to the trace unit.
0x3214	CYCLES_ARB_PENDING_CME	**Cycles SME2 unit arbitration pending** The counter counts each PE cycle counted by CPU_CYCLES where the CPU is in waiting for arbitration while attempting to access the SME2 unit. If can be due an initial arbitration request or contention. **Note:** This event is not exported to the trace unit.
0x3215	CYCLES_CME_RECONNECT_PENDING	**Cycles SME2 unit reconnect pending** The counter counts each PE cycle counted by CYCLES_ARB_PENDING_CME where the CPU is in waiting for arbitration due to contention, when it was previously arbitrated to a SME2 unit but arbitration was granted to another CPU. **Note:** This event is not exported to the trace unit.
0x3216	ARB_CME_COUNT	**SME2 unit arbitration requests** The counter counts the number of times an arbitration to SME2 unit is requested, either an initial request, or a reconnect request due to contention. **Note:** This event is not exported to the trace unit.
0x3217	RECONNECT_CME_COUNT	**SME2 unit reconnect requests** The counter counts the number of times the CPU needs to request arbitration following a disconnect due to contention. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x3218	OP_CME_ISSUE	**SME operation issued** The counter counts each operation that was issued to a streaming mode compute unit. The definition of which operations are issued to an SME2 unit is IMPLEMENTATION DEFINED. The maximum value by which the counter could increment by in a single cycle is IMPLEMENTATION DEFINED. **Note:** This event is not exported to the trace unit.
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. **Note:** This event is not exported to the trace unit.
0x321A	SSVE_SPEC	**Operation speculatively executed, Streaming SVE** The counter counts each operation counted by SVE_SPEC specifically in Streaming mode. **Note:** This event is not exported to the trace unit.
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. **Note:** This event is not exported to the trace unit.
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. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
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. **Note:** This event is not exported to the trace unit.
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. **Note:** This event is not exported to the trace unit.
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. **Note:** This event is not exported to the trace unit.

17.3 Performance monitors interrupts

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

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

For more information, see *Performance Monitors Extension support* in the *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.

17.4 External register access permissions

The C1-Ultra core supports access to the Performance Monitoring Unit (PMU) registers from the system register interface and a memory-mapped interface.

Access to a register depends on:

- Whether the core is powered up
- The state of the OS Lock

- The state of External Performance Monitors Access Disable

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

17.5 AArch64 Performance Monitors registers

The following summary table provides an overview of all PMU registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 17-3: PMU registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMINTENSET_EL1	3	0	C9	C14	1	See individual bit resets.	64-bit	Performance Monitors Interrupt Enable Set Register
PMINTENCLR_EL1	3	0	C9	C14	2	See individual bit resets.	64-bit	Performance Monitors Interrupt Enable Clear Register
PMMIR_EL1	3	0	C9	C14	6	0x000000000000000A	64-bit	Performance Monitors Machine Identification Register
PMCR_EL0	3	3	C9	C12	0	See individual bit resets.	64-bit	Performance Monitors Control Register
PMCNTENSET_EL0	3	3	C9	C12	1	See individual bit resets.	64-bit	Performance Monitors Count Enable Set Register
PMCNTENCLR_EL0	3	3	C9	C12	2	See individual bit resets.	64-bit	Performance Monitors Count Enable Clear Register
PMOVSCLR_EL0	3	3	C9	C12	3	See individual bit resets.	64-bit	Performance Monitors Overflow Flag Status Clear Register
PMSWINC_EL0	3	3	C9	C12	4	See individual bit resets.	64-bit	Performance Monitors Software Increment Register
PMSELR_EL0	3	3	C9	C12	5	See individual bit resets.	64-bit	Performance Monitors Event Counter Selection Register
PMCEID0_EL0	3	3	C9	C12	6	See individual bit resets.	64-bit	Performance Monitors Common Event Identification Register 0
PMCEID1_EL0	3	3	C9	C12	7	See individual bit resets.	64-bit	Performance Monitors Common Event Identification Register 1
PMCCNTR_EL0	3	3	C9	C13	0	See individual bit resets.	64-bit	Performance Monitors Cycle Count Register
PMXEVTYPER_EL0	3	3	C9	C13	1	See individual bit resets.	64-bit	Performance Monitors Selected Event Type Register
PMXEVCNTR_EL0	3	3	C9	C13	2	See individual bit resets.	64-bit	Performance Monitors Selected Event Count Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMUSERENR_EL0	3	3	C9	C14	0	See individual bit resets.	64-bit	Performance Monitors User Enable Register
PMOVSSET_EL0	3	3	C9	C14	3	See individual bit resets.	64-bit	Performance Monitors Overflow Flag Status Set Register
PMEVCNTR0_EL0	3	3	C14	C8	0	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR1_EL0	3	3	C14	C8	1	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR2_EL0	3	3	C14	C8	2	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR3_EL0	3	3	C14	C8	3	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR4_EL0	3	3	C14	C8	4	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR5_EL0	3	3	C14	C8	5	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR6_EL0	3	3	C14	C8	6	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR7_EL0	3	3	C14	C8	7	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR8_EL0	3	3	C14	C9	0	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR9_EL0	3	3	C14	C9	1	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR10_EL0	3	3	C14	C9	2	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR11_EL0	3	3	C14	C9	3	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR12_EL0	3	3	C14	C9	4	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR13_EL0	3	3	C14	C9	5	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR14_EL0	3	3	C14	C9	6	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR15_EL0	3	3	C14	C9	7	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR16_EL0	3	3	C14	C10	0	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR17_EL0	3	3	C14	C10	1	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR18_EL0	3	3	C14	C10	2	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR19_EL0	3	3	C14	C10	3	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR20_EL0	3	3	C14	C10	4	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR21_EL0	3	3	C14	C10	5	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR22_EL0	3	3	C14	C10	6	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR23_EL0	3	3	C14	C10	7	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR24_EL0	3	3	C14	C11	0	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR25_EL0	3	3	C14	C11	1	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR26_EL0	3	3	C14	C11	2	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR27_EL0	3	3	C14	C11	3	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR28_EL0	3	3	C14	C11	4	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR29_EL0	3	3	C14	C11	5	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR30_EL0	3	3	C14	C11	6	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVTYPER0_EL0	3	3	C14	C12	0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER1_EL0	3	3	C14	C12	1	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER2_EL0	3	3	C14	C12	2	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER3_EL0	3	3	C14	C12	3	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER4_EL0	3	3	C14	C12	4	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER5_EL0	3	3	C14	C12	5	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER6_EL0	3	3	C14	C12	6	See individual bit resets.	64-bit	Performance Monitors Event Type Registers

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMEVTYPER7_EL0	3	3	C14	C12	7	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER8_EL0	3	3	C14	C13	0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER9_EL0	3	3	C14	C13	1	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER10_EL0	3	3	C14	C13	2	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER11_EL0	3	3	C14	C13	3	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER12_EL0	3	3	C14	C13	4	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER13_EL0	3	3	C14	C13	5	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER14_EL0	3	3	C14	C13	6	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER15_EL0	3	3	C14	C13	7	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER16_EL0	3	3	C14	C14	0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER17_EL0	3	3	C14	C14	1	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER18_EL0	3	3	C14	C14	2	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER19_EL0	3	3	C14	C14	3	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER20_EL0	3	3	C14	C14	4	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER21_EL0	3	3	C14	C14	5	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER22_EL0	3	3	C14	C14	6	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER23_EL0	3	3	C14	C14	7	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER24_EL0	3	3	C14	C15	0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER25_EL0	3	3	C14	C15	1	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER26_EL0	3	3	C14	C15	2	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER27_EL0	3	3	C14	C15	3	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER28_EL0	3	3	C14	C15	4	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER29_EL0	3	3	C14	C15	5	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER30_EL0	3	3	C14	C15	6	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMCCFILTR_EL0	3	3	C14	C15	7	See individual bit resets.	64-bit	Performance Monitors Cycle Count Filter Register

17.6 External PMU registers summary

The following summary table provides an overview of all memory-mapped PMU registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 17-4: PMU registers summary

Offset	Name	Reset	Width	Description
0x0	PMEVCNTR0_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x8	PMEVCNTR1_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x10	PMEVCNTR2_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x18	PMEVCNTR3_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x20	PMEVCNTR4_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x28	PMEVCNTR5_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x30	PMEVCNTR6_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x38	PMEVCNTR7_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x40	PMEVCNTR8_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x48	PMEVCNTR9_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x50	PMEVCNTR10_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x58	PMEVCNTR11_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x60	PMEVCNTR12_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x68	PMEVCNTR13_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x70	PMEVCNTR14_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x78	PMEVCNTR15_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x80	PMEVCNTR16_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x88	PMEVCNTR17_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x90	PMEVCNTR18_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x98	PMEVCNTR19_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xA0	PMEVCNTR20_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xA8	PMEVCNTR21_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xB0	PMEVCNTR22_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xB8	PMEVCNTR23_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xC0	PMEVCNTR24_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xC8	PMEVCNTR25_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xD0	PMEVCNTR26_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xD8	PMEVCNTR27_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xE0	PMEVCNTR28_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xE8	PMEVCNTR29_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xF0	PMEVCNTR30_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x0F8	PMCCNTR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter
0x0FC	PMCCNTR_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter
0x200	PMPCSR [31:0]	See individual bit resets.	32-bit	Program Counter Sample Register
0x204	PMPCSR [63:32]	See individual bit resets.	32-bit	Program Counter Sample Register
0x220	PMPCSR [31:0]	See individual bit resets.	32-bit	Program Counter Sample Register
0x224	PMPCSR [63:32]	See individual bit resets.	32-bit	Program Counter Sample Register
0x208	PMCID1SR	See individual bit resets.	32-bit	CONTEXTIDR_EL1 Sample Register
0x228	PMCID1SR	See individual bit resets.	32-bit	CONTEXTIDR_EL1 Sample Register

Offset	Name	Reset	Width	Description
0x20C	PMVIDSR	See individual bit resets.	32-bit	VMID Sample Register
0x22C	PMCID2SR	See individual bit resets.	32-bit	CONTEXTIDR_EL2 Sample Register
0x400	PMEVTYPER0_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA00	PMEVTYPER0_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x404	PMEVTYPER1_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA04	PMEVTYPER1_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x408	PMEVTYPER2_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA08	PMEVTYPER2_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x40C	PMEVTYPER3_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA0C	PMEVTYPER3_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x410	PMEVTYPER4_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA10	PMEVTYPER4_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x414	PMEVTYPER5_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA14	PMEVTYPER5_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x418	PMEVTYPER6_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA18	PMEVTYPER6_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x41C	PMEVTYPER7_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA1C	PMEVTYPER7_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x420	PMEVTYPER8_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA20	PMEVTYPER8_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x424	PMEVTYPER9_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA24	PMEVTYPER9_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x428	PMEVTYPER10_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA28	PMEVTYPER10_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x42C	PMEVTYPER11_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA2C	PMEVTYPER11_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x430	PMEVTYPER12_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA30	PMEVTYPER12_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x434	PMEVTYPER13_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA34	PMEVTYPER13_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x438	PMEVTYPER14_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA38	PMEVTYPER14_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x43C	PMEVTYPER15_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA3C	PMEVTYPER15_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x440	PMEVTYPER16_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA40	PMEVTYPER16_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x444	PMEVTYPER17_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA44	PMEVTYPER17_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x448	PMEVTYPER18_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA48	PMEVTYPER18_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers

Offset	Name	Reset	Width	Description
0x44C	PMEVTYPER19_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA4C	PMEVTYPER19_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x450	PMEVTYPER20_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA50	PMEVTYPER20_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x454	PMEVTYPER21_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA54	PMEVTYPER21_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x458	PMEVTYPER22_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA58	PMEVTYPER22_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x45C	PMEVTYPER23_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA5C	PMEVTYPER23_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x460	PMEVTYPER24_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA60	PMEVTYPER24_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x464	PMEVTYPER25_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA64	PMEVTYPER25_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x468	PMEVTYPER26_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA68	PMEVTYPER26_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x46C	PMEVTYPER27_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA6C	PMEVTYPER27_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x470	PMEVTYPER28_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA70	PMEVTYPER28_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x474	PMEVTYPER29_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA74	PMEVTYPER29_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x478	PMEVTYPER30_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA78	PMEVTYPER30_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x47C	PMCCFILTR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter Filter Register
0xA7C	PMCCFILTR_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter Filter Register
0x600	PMPCSSR	See individual bit resets.	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x60C	PMCID2SSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register
0x610	PMSSSR	See individual bit resets.	32-bit	PMU Snapshot Status Register
0x618	PMCCNTSR	See individual bit resets.	64-bit	PMU Cycle Counter Snapshot Register
0x620	PMEVCNTSR0	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x628	PMEVCNTSR1	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x630	PMEVCNTSR2	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x638	PMEVCNTSR3	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x640	PMEVCNTSR4	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x648	PMEVCNTSR5	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x650	PMEVCNTSR6	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x658	PMEVCNTSR7	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x660	PMEVCNTSR8	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register

Offset	Name	Reset	Width	Description
0x668	PMEVCNTSR9	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x670	PMEVCNTSR10	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x678	PMEVCNTSR11	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x680	PMEVCNTSR12	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x688	PMEVCNTSR13	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x690	PMEVCNTSR14	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x698	PMEVCNTSR15	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A0	PMEVCNTSR16	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A8	PMEVCNTSR17	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B0	PMEVCNTSR18	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B8	PMEVCNTSR19	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6C0	PMEVCNTSR20	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6C8	PMEVCNTSR21	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6D0	PMEVCNTSR22	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6D8	PMEVCNTSR23	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6E0	PMEVCNTSR24	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6E8	PMEVCNTSR25	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F0	PMEVCNTSR26	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F8	PMEVCNTSR27	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x700	PMEVCNTSR28	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x708	PMEVCNTSR29	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x710	PMEVCNTSR30	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0xC00	PMCNTENSET_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Count Enable Set Register
0xC20	PMCNTENCLR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Count Enable Clear Register
0xC40	PMINTENSET_EL1 [31:0]	See individual bit resets.	32-bit	Performance Monitors Interrupt Enable Set Register
0xC60	PMINTENCLR_EL1 [31:0]	See individual bit resets.	32-bit	Performance Monitors Interrupt Enable Clear Register
0xC80	PMOVSCLR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Overflow Flag Status Clear register
0xCA0	PMSWINC_EL0	See individual bit resets.	32-bit	Performance Monitors Software Increment Register
0xCC0	PMOVSSET_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Overflow Flag Status Set Register
0xE00	PMCFGR [31:0]	See individual bit resets.	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_EL0	See individual bit resets.	32-bit	Performance Monitors Control Register
0xE08	PMIIDR	See individual bit resets.	32-bit	Performance Monitors Implementation Identification Register
0xE20	PMCEID0	`0x7FFF6F3F`	32-bit	Performance Monitors Common Event Identification register 0
0xE24	PMCEID1	`0xFEF2AE7F`	32-bit	Performance Monitors Common Event Identification register 1
0xE28	PMCEID2	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 2
0xE2C	PMCEID3	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 3
0xE30	PMSSCR	See individual bit resets.	32-bit	PMU Snapshot Capture Register
0xE40	PMMIR [31:0]	`0x0000000A`	32-bit	Performance Monitors Machine Identification Register
0xFA8	PMDEVAFF0	See individual bit resets.	32-bit	Performance Monitors Device Affinity register 0
0xFAC	PMDEVAFF1	See individual bit resets.	32-bit	Performance Monitors Device Affinity register 1

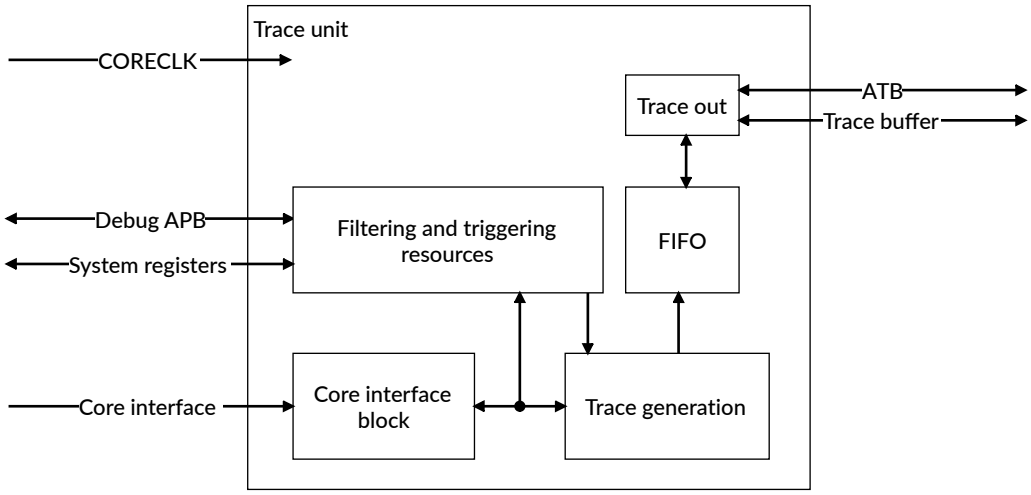
Offset	Name	Reset	Width	Description
0xFB0	PMLAR	See individual bit resets.	32-bit	Performance Monitors Lock Access Register
0xFB4	PMLSR	See individual bit resets.	32-bit	Performance Monitors Lock Status Register
0xFB8	PMAUTHSTATUS	See individual bit resets.	32-bit	Performance Monitors Authentication Status register
0xFBC	PMDEVARCH	0x47702A16	32-bit	Performance Monitors Device Architecture register
0xFC8	PMDEVID	0x00000001	32-bit	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	0x00000016	32-bit	Performance Monitors Device Type register
0xFD0	PMPIDR4	0x00000004	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	0x0000008C	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	0x000000BD	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	0x0000001B	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	0x00000000	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	0x0000000D	32-bit	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	0x00000090	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	0x00000005	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	0x000000B1	32-bit	Performance Monitors Component Identification Register 3

18. Embedded Trace Extension support

The C1-Ultra core implements the Embedded Trace Extension (ETE). The trace unit performs real-time instruction flow tracing based on the ETE. The trace unit is a CoreSight component and is an integral part of the Arm real-time debug solution.

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

Figure 18-1: Trace unit components



Core interface

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

Trace generation

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

Filtering and triggering resources

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

FIFO

The trace unit generates trace in a highly compressed form. The First In First Out (FIFO) enables trace bursts to be flattened out. When the FIFO is full, the FIFO signals an overflow. The trace

generation logic does not generate any new trace until the FIFO is emptied. This behavior causes a gap in the trace when viewed in the debugger.

Trace out

Trace from the FIFO is output on the AMBA® Trace Bus (ATB) interface or to the trace buffer.

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

18.1 Trace unit resources

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

The following table shows the trace unit resources, and indicates which of these resources the C1-Ultra core implements.

Table 18-1: Trace unit resources implemented

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

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

18.2 Trace unit generation options

The C1-Ultra core trace unit implements a set of generation options.

The following table shows the trace generation options that are implemented in the C1-Ultra core trace unit.

Table 18-2: Trace unit generation options implemented

Description	Configuration
Instruction address size in bytes	8
Data address size in bytes	0, because the Embedded Trace Extension (ETE) does not implement data tracing
Data value size in bytes	0, because the ETE does not implement data tracing
Virtual Machine ID size in bytes	4
Context ID size in bytes	4
Support for conditional instruction tracing	Not implemented
Support for tracing of data	Not implemented
Support for tracing of load and store instructions as P0 elements	Not implemented
Support for cycle counting in the instruction trace	Implemented
Support for branch broadcast tracing	Implemented
Number of events that are supported in the trace	4
Return stack support	Implemented
Tracing of SError exception support	Implemented
Instruction trace cycle counting minimum threshold	4
Size of Trace ID	7 bits
Synchronization period support	Read/write
Global timestamp size	64 bits
Number of cores available for tracing	1
ATB trigger support	Implemented
Low-power behavior override	Not implemented
Stall control support	Not implemented
Support for overflow avoidance	Not implemented
Support for using CONTEXTIDR_EL2 in Virtual Machine Identifier (VMID) comparator	Implemented

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

18.3 Reset the trace unit

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

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

18.4 Program and read the trace unit registers

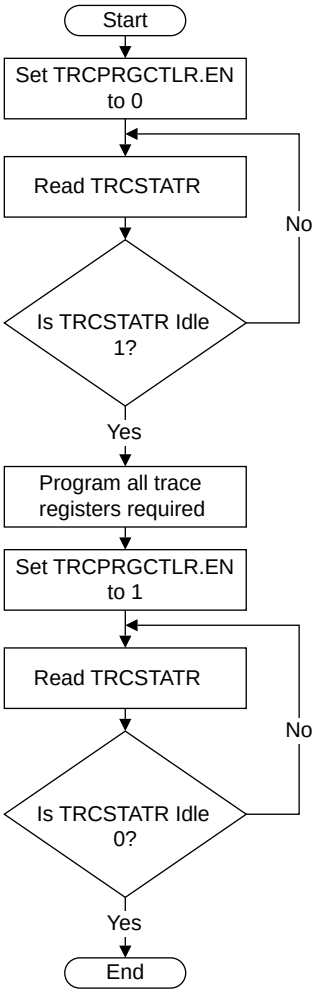
You program and read the trace unit registers using either the Debug Advanced Peripheral Bus (APB) interface or the System register interface.

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

For more information about the Programming Control Register, TRCPRGCTLR, and the Trace Status Register, TRCSTATR, see the *Arm® Architecture Reference Manual for A-profile architecture*.

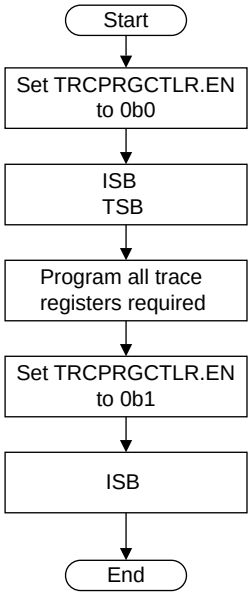
The following figure shows the flow for programming trace unit registers using the Debug APB interface.

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



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

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



18.5 Trace unit register interfaces

The C1-Ultra core supports an Advanced Peripheral Bus (APB) memory-mapped interface and a system register interface to trace unit registers.

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

18.6 Interaction with the Performance Monitoring Unit and Debug

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

Interaction with the PMU block

The C1-Ultra core includes a PMU block that allows for events, such as cache misses and executed instructions, to be counted over time.

The PMU and trace unit function together.

Use of PMU events by the trace unit

The PMU architectural events are available to the trace unit through the extended input facility. For more information about PMU events, see the *Arm® Architecture Reference Manual for A-profile architecture*.

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

Related information

17. Performance Monitors Extension support on page 125

18.7 Embedded Trace Extension events

The C1-Ultra core trace unit collects events from other units in the design and uses numbers to reference these events.

Some Performance Monitoring Unit (PMU) events are exported to the trace unit and are either not counted in or not visible to the PMU.

The exported events are listed in the table in 17.1 Common PMU events on page 125.

The events listed in the following table are exported to the Embedded Trace Extension (ETE), but cannot be counted in the PMU.

Table 18-3: ETE events

Event number	Event mnemonic	Description
0x400D	PMU_OVFS	**PMU overflow, counters accessible to EL1 and EL0** This event is generated each time an event causes a PMEVCTNR<n>_EL1 counter overflow when PMINTENSET_EL1[n] is set to 1, for each implemented PMU counter n in the range 0 <= n < UInt(MDCR_EL2.HPMN), and the Cycle Counter (n = 31). **Note:** This event is exported to the trace unit, but cannot be counted in the PMU.
0x400E	TRB_TRIG	**Trace buffer Trigger Event** This event is generated when a Trace Buffer Extension Trigger Event occurs. **Note:** This event is exported to the trace unit, but cannot be counted in the PMU.

Event number	Event mnemonic	Description
0x400F	PMU_HOVFS	**PMU overflow, counters reserved for use by EL2** This event is generated each time an event causes a PMEVCTNR<n>_EL1 counter overflow when PMINTENSET_EL1[n] is set to 1, for each implemented PMU counter n in the range UInt(MDCR_EL2.HPMN) <= n < UInt(PMCR_EL0.N). This event is not transmitted to a PE Trace Unit while TRCFR_EL2.E2TRE == 0b0. **Note:** This event is exported to the trace unit, but cannot be counted in the PMU.

18.8 AArch64 Trace unit registers

The following summary table provides an overview of all Trace registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 18-4: Trace registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCTRACEIDR	2	1	C0	C0	1	See individual bit resets.	64-bit	Trace ID Register
TRCVICTLR	2	1	C0	C0	2	See individual bit resets.	64-bit	Trace ViewInst Main Control Register
TRCSEQEVR0	2	1	C0	C0	4	See individual bit resets.	64-bit	Trace Sequencer State Transition Control Register <n>
TRCCNTRLDVR0	2	1	C0	C0	5	See individual bit resets.	64-bit	Trace Counter Reload Value Register <n>
TRCIDR8	2	1	C0	C0	6	0x0000000000000000	64-bit	Trace ID Register 8
TRCIMSPEC0	2	1	C0	C0	7	See individual bit resets.	64-bit	Trace IMP DEF Register 0
TRCPRGCTLR	2	1	C0	C1	0	See individual bit resets.	64-bit	Trace Programming Control Register
TRCVIIECTLR	2	1	C0	C1	2	See individual bit resets.	64-bit	Trace ViewInst Include/Exclude Control Register
TRCSEQEVR1	2	1	C0	C1	4	See individual bit resets.	64-bit	Trace Sequencer State Transition Control Register <n>
TRCCNTRLDVR1	2	1	C0	C1	5	See individual bit resets.	64-bit	Trace Counter Reload Value Register <n>
TRCIDR9	2	1	C0	C1	6	0x0000000000000000	64-bit	Trace ID Register 9
TRCVISSCTLR	2	1	C0	C2	2	See individual bit resets.	64-bit	Trace ViewInst Start/Stop Control Register
TRCSEQEVR2	2	1	C0	C2	4	See individual bit resets.	64-bit	Trace Sequencer State Transition Control Register <n>
TRCIDR10	2	1	C0	C2	6	0x0000000000000000	64-bit	Trace ID Register 10
TRCSTATR	2	1	C0	C3	0	See individual bit resets.	64-bit	Trace Status Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCIDR11	2	1	C0	C3	6	0x0000000000000000	64-bit	Trace ID Register 11
TRCCONFIGR	2	1	C0	C4	0	See individual bit resets.	64-bit	Trace Configuration Register
TRCCNTCTLR0	2	1	C0	C4	5	See individual bit resets.	64-bit	Trace Counter Control Register <n>
TRCIDR12	2	1	C0	C4	6	0x0000000000000000	64-bit	Trace ID Register 12
TRCCNTCTLR1	2	1	C0	C5	5	See individual bit resets.	64-bit	Trace Counter Control Register <n>
TRCIDR13	2	1	C0	C5	6	0x0000000000000000	64-bit	Trace ID Register 13
TRCAUXCTLR	2	1	C0	C6	0	See individual bit resets.	64-bit	Trace Auxiliary Control Register
TRCSEQRSTEVR	2	1	C0	C6	4	See individual bit resets.	64-bit	Trace Sequencer Reset Control Register
TRCSEQSTR	2	1	C0	C7	4	See individual bit resets.	64-bit	Trace Sequencer State Register
TRCEVENTCTL0R	2	1	C0	C8	0	See individual bit resets.	64-bit	Trace Event Control 0 Register
TRCEXTINSELR0	2	1	C0	C8	4	See individual bit resets.	64-bit	Trace External Input Select Register <n>
TRCCNTVR0	2	1	C0	C8	5	See individual bit resets.	64-bit	Trace Counter Value Register <n>
TRCIDR0	2	1	C0	C8	7	0x0000000028800EA1	64-bit	Trace ID Register 0
TRCEVENTCTL1R	2	1	C0	C9	0	See individual bit resets.	64-bit	Trace Event Control 1 Register
TRCEXTINSELR1	2	1	C0	C9	4	See individual bit resets.	64-bit	Trace External Input Select Register <n>
TRCCNTVR1	2	1	C0	C9	5	See individual bit resets.	64-bit	Trace Counter Value Register <n>
TRCIDR1	2	1	C0	C9	7	0x000000004100FFF0	64-bit	Trace ID Register 1
TRCRSR	2	1	C0	C10	0	See individual bit resets.	64-bit	Trace Resources Status Register
TRCEXTINSELR2	2	1	C0	C10	4	See individual bit resets.	64-bit	Trace External Input Select Register <n>
TRCIDR2	2	1	C0	C10	7	0x00000000C0001088	64-bit	Trace ID Register 2
TRCEXTINSELR3	2	1	C0	C11	4	See individual bit resets.	64-bit	Trace External Input Select Register <n>
TRCIDR3	2	1	C0	C11	7	0x00000000017F0004	64-bit	Trace ID Register 3
TRCTSCTLR	2	1	C0	C12	0	See individual bit resets.	64-bit	Trace Timestamp Control Register
TRCIDR4	2	1	C0	C12	7	0x0000000011170004	64-bit	Trace ID Register 4
TRCSYNCPR	2	1	C0	C13	0	See individual bit resets.	64-bit	Trace Synchronization Period Register
TRCIDR5	2	1	C0	C13	7	0x00000000284709FF	64-bit	Trace ID Register 5
TRCCCCTLR	2	1	C0	C14	0	See individual bit resets.	64-bit	Trace Cycle Count Control Register
TRCIDR6	2	1	C0	C14	7	0x0000000000000000	64-bit	Trace ID Register 6
TRCBBCTLR	2	1	C0	C15	0	See individual bit resets.	64-bit	Trace Branch Broadcast Control Register
TRCIDR7	2	1	C0	C15	7	0x0000000000000000	64-bit	Trace ID Register 7
TRCSSCCR0	2	1	C1	C0	2	See individual bit resets.	64-bit	Trace Single-shot Comparator Control Register <n>
TRCOSLSR	2	1	C1	C1	4	See individual bit resets.	64-bit	Trace OS Lock Status Register
TRCRSCTLR2	2	1	C1	C2	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR3	2	1	C1	C3	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR4	2	1	C1	C4	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR5	2	1	C1	C5	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR6	2	1	C1	C6	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR7	2	1	C1	C7	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR8	2	1	C1	C8	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCSSCSR0	2	1	C1	C8	2	See individual bit resets.	64-bit	Trace Single-shot Comparator Control Status Register <n>

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCRSCTLR9	2	1	C1	C9	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR10	2	1	C1	C10	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR11	2	1	C1	C11	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR12	2	1	C1	C12	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR13	2	1	C1	C13	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR14	2	1	C1	C14	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR15	2	1	C1	C15	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCACVR0	2	1	C2	C0	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR0	2	1	C2	C0	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR1	2	1	C2	C2	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR1	2	1	C2	C2	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR2	2	1	C2	C4	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR2	2	1	C2	C4	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR3	2	1	C2	C6	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR3	2	1	C2	C6	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR4	2	1	C2	C8	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR4	2	1	C2	C8	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR5	2	1	C2	C10	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR5	2	1	C2	C10	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR6	2	1	C2	C12	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR6	2	1	C2	C12	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR7	2	1	C2	C14	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR7	2	1	C2	C14	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCCIDCVR0	2	1	C3	C0	0	See individual bit resets.	64-bit	Trace Context Identifier Comparator Value Registers <n>
TRCVMIDCVR0	2	1	C3	C0	1	See individual bit resets.	64-bit	Trace Virtual Context Identifier Comparator Value Register <n>
TRCCIDCCTLR0	2	1	C3	C0	2	See individual bit resets.	64-bit	Trace Context Identifier Comparator Control Register 0
TRCVMIDCCTLR0	2	1	C3	C2	2	See individual bit resets.	64-bit	Trace Virtual Context Identifier Comparator Control Register 0
TRCDEVID	2	1	C7	C2	7	0x0000000000000000	64-bit	Trace Device Configuration Register
TRCCLAIMSET	2	1	C7	C8	6	See individual bit resets.	64-bit	Trace Claim Tag Set Register
TRCCLAIMCLR	2	1	C7	C9	6	See individual bit resets.	64-bit	Trace Claim Tag Clear Register
TRCAUTHSTATUS	2	1	C7	C14	6	See individual bit resets.	64-bit	Trace Authentication Status Register
TRCDEVARCH	2	1	C7	C15	6	0x0000000047715A13	64-bit	Trace Device Architecture Register

18.9 External ETE registers

The following summary table provides an overview of all memory-mapped ETE registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 18-5: ETE registers summary

Offset	Name	Reset	Width	Description
0x018	TRCAUXCTLR	See individual bit resets.	32-bit	Trace Auxiliary Control Register
0x180	TRCIDR8	0x00000000	32-bit	Trace ID Register 8
0x184	TRCIDR9	0x00000000	32-bit	Trace ID Register 9
0x188	TRCIDR10	0x00000000	32-bit	Trace ID Register 10
0x18C	TRCIDR11	0x00000000	32-bit	Trace ID Register 11
0x190	TRCIDR12	0x00000000	32-bit	Trace ID Register 12
0x194	TRCIDR13	0x00000000	32-bit	Trace ID Register 13
0x1C0	TRCIMSPEC0	See individual bit resets.	32-bit	Trace IMP DEF Register 0
0x1E0	TRCIDR0	0x28800EA1	32-bit	Trace ID Register 0
0x1E4	TRCIDR1	0x4100FFF0	32-bit	Trace ID Register 1
0x1E8	TRCIDR2	0xC0001088	32-bit	Trace ID Register 2
0x1EC	TRCIDR3	0x017F0004	32-bit	Trace ID Register 3
0x1F0	TRCIDR4	0x11170004	32-bit	Trace ID Register 4
0x1F4	TRCIDR5	0x284709FF	32-bit	Trace ID Register 5
0x1F8	TRCIDR6	0x00000000	32-bit	Trace ID Register 6
0x1FC	TRCIDR7	0x00000000	32-bit	Trace ID Register 7
0xF00	TRCITCTRL	See individual bit resets.	32-bit	Trace Integration Mode Control Register
0xFA0	TRCCLAIMSET	0xFFFFFFFF	32-bit	Trace Claim Tag Set Register
0xFA4	TRCCLAIMCLR	0x00000000	32-bit	Trace Claim Tag Clear Register
0xFBC	TRCDEVARCH	0x47715A13	32-bit	Trace Device Architecture Register
0xFC0	TRCDEVID2	0x00000000	32-bit	Trace Device Configuration Register 2
0xFC4	TRCDEVID1	0x00000000	32-bit	Trace Device Configuration Register 1
0xFC8	TRCDEVID	0x00000000	32-bit	Trace Device Configuration Register
0xFCC	TRCDEVTYPE	0x00000013	32-bit	Trace Device Type Register
0xFD0	TRCPIDR4	See individual bit resets.	32-bit	Trace Peripheral Identification Register 4
0xFD4	TRCPIDR5	0x00000000	32-bit	Trace Peripheral Identification Register 5
0xFD8	TRCPIDR6	0x00000000	32-bit	Trace Peripheral Identification Register 6

Offset	Name	Reset	Width	Description
0xFDC	TRCPIDR7	0x00000000	32-bit	Trace Peripheral Identification Register 7
0xFE0	TRCPIDR0	0x0000008C	32-bit	Trace Peripheral Identification Register 0
0xFE4	TRCPIDR1	0x000000BD	32-bit	Trace Peripheral Identification Register 1
0xFE8	TRCPIDR2	0x0000001B	32-bit	Trace Peripheral Identification Register 2
0xFEC	TRCPIDR3	0x00000000	32-bit	Trace Peripheral Identification Register 3
0xFF0	TRCCIDR0	0x0000000D	32-bit	Trace Component Identification Register 0
0xFF4	TRCCIDR1	0x00000090	32-bit	Trace Component Identification Register 1
0xFF8	TRCCIDR2	0x00000005	32-bit	Trace Component Identification Register 2
0xFFC	TRCCIDR3	0x000000B1	32-bit	Trace Component Identification Register 3

19. Trace Buffer Extension support

The C1-Ultra core implements the TRace Buffer Extension (TRBE). The TRBE writes the program flow trace generated by the trace unit directly to memory. The TRBE is programmed through System registers.

When enabled, the TRBE can:

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

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

19.1 Program and read the trace buffer registers

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

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

For more information on the TRBE register behaviors and access mechanisms, see the *Arm® Architecture Reference Manual for A-profile architecture*

19.2 Trace buffer register interface

The C1-Ultra core supports a System register interface to TRace Buffer Extension (TRBE) registers.

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

20. Activity Monitors Extension support

The C1-Ultra core implements the Activity Monitors Extension to the Arm®v8.4-A architecture. Activity monitoring has features similar to performance monitoring features, but is intended for system management use whereas performance monitoring is aimed at user and debug applications.

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

The C1-Ultra core implements ten counters in two groups, each of which is a 64-bit counter that counts a fixed event. Group 0 has four counters 00-03, and Group 1 has six counters 10-15.

20.1 Activity monitors access

The C1-Ultra core supports access to activity monitors from the System register interface and supports read-only memory-mapped access using the utility bus interface.

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

Access enable bit

There are multiple access enable bits associated with the Activity Monitors System registers:

- AMUSERENR_EL0.EN controls access from EL0 to the Activity Monitors System registers.
- CPTR_EL2.TAM controls access from EL0 and EL1 to the Activity Monitors System registers.
- CPTR_EL3.TAM controls access from EL0, EL1, and EL2 to the Activity Monitors Extension System registers.

For a detailed description of access controls for the registers, see the *Arm® Architecture Reference Manual for A-profile architecture*.

System register access

The activity monitors are accessible using the `MRS` and `MSR` instructions.

External memory-mapped access

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

The base address for Activity Monitoring Unit (AMU) registers on the utility bus interface is `0x<n>90000`, where n is the C1-Ultra core instance number in the C1-DSU cluster.

These registers are treated as RAZ/WI if the register is marked as Reserved or if the register is accessed in the wrong Security state.

20.2 Activity monitors counters

The C1-Ultra core implements four activity monitors counters, 0-3, and six auxiliary counters, 10-15, 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 a `WFI`, `WFE`, `WFIT`, or `WFET` instruction stops the clock.
- Events 0-3 and auxiliary events 10-15 are fixed, and the AMEVTYPER0<n>_EL0 and AMEVTYPER1<n>_EL0 evtCount bits are read-only.
- The activity monitor counters are reset to zero on a Cold reset of the power domain of the core. When the core is not in reset, activity monitoring is available.

20.3 Activity monitors events

Activity monitors events in the C1-Ultra core are either fixed or programmable, and they map to the activity monitors counters.

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

Table 20-1: Mapping of counters to fixed events

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

Activity monitor counter <n>	Event	Event number	Description
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
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.
AMEVCNTR14	STALL_BACKEND_BUSY_CME	0x3200	The CPU is stalling because of SME2 backpressure, for any reason. Typical reasons for stalling include: • The core arbitrated in the C1-SME2, but the core sends instructions faster than the SME2 can execute them. • The core is waiting for arbitration and stalls instruction execution.
AMEVCNTR15	STALL_BACKEND_BUSY_CME_ARB	0x3202	The core is stalling because of SME2 backpressure, waiting for arbitration. This event is a subset of STALL_BACKEND_CME event. This event is set when the CPU stalls because of SME2 contention, waiting for arbitration.

Related information

4.5.1 Maximum Power Mitigation Mechanism on page 52

20.4 AArch64 AMU registers summary

The following summary table provides an overview of all AMU registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 20-2: AMU registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMCR_EL0	3	3	C13	C2	0	See individual bit resets.	64-bit	Activity Monitors Control Register
AMCFGR_EL0	3	3	C13	C2	1	0x0000000011003F09	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	3	3	C13	C2	2	0x0000000000000604	64-bit	Activity Monitors Counter Group Configuration Register
AMUSERENR_EL0	3	3	C13	C2	3	See individual bit resets.	64-bit	Activity Monitors User Enable Register
AMCNTENCLR0_EL0	3	3	C13	C2	4	See individual bit resets.	64-bit	Activity Monitors Count Enable Clear Register 0
AMCNTENSET0_EL0	3	3	C13	C2	5	See individual bit resets.	64-bit	Activity Monitors Count Enable Set Register 0
AMCNTENCLR1_EL0	3	3	C13	C3	0	See individual bit resets.	64-bit	Activity Monitors Count Enable Clear Register 1
AMCNTENSET1_EL0	3	3	C13	C3	1	See individual bit resets.	64-bit	Activity Monitors Count Enable Set Register 1
AMEVCNTR00_EL0	3	3	C13	C4	0	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR01_EL0	3	3	C13	C4	1	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR02_EL0	3	3	C13	C4	2	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR03_EL0	3	3	C13	C4	3	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
AMEVTYPER00_EL0	3	3	C13	C6	0	0x0000000000000011	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	3	C13	C6	1	0x0000000000004004	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	3	C13	C6	2	0x0000000000000008	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	3	C13	C6	3	0x0000000000004005	64-bit	Activity Monitors Event Type Registers 0
AMEVCNTR10_EL0	3	3	C13	C12	0	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR11_EL0	3	3	C13	C12	1	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR12_EL0	3	3	C13	C12	2	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR13_EL0	3	3	C13	C12	3	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR14_EL0	3	3	C13	C12	4	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR15_EL0	3	3	C13	C12	5	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
AMEVTYPER10_EL0	3	3	C13	C14	0	0x0000000000000300	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	3	C13	C14	1	0x0000000000000301	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	3	C13	C14	2	0x0000000000000302	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER13_EL0	3	3	C13	C14	3	0x0000000000000310	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER14_EL0	3	3	C13	C14	4	0x0000000000003200	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER15_EL0	3	3	C13	C14	5	0x0000000000003202	64-bit	Activity Monitors Event Type Registers 1

20.5 External AMU registers

The following summary table provides an overview of all 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.

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 20-3: 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
0x120	AMEVCNTR14 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
0x128	AMEVCNTR15 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
0x400	AMEVTYPER00	0x00000011	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	0x00004004	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	0x00000008	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	0x00004005	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
0x490	AMEVTYPER14	0x00003200	32-bit	Activity Monitors Event Type Registers 1
0x494	AMEVTYPER15	0x00003202	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	0x00000604	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	0x11003F09	32-bit	Activity Monitors Configuration Register
0xE04	AMCR	See individual bit resets.	32-bit	Activity Monitors Control Register

Offset	Name	Reset	Width	Description
0xE08	AMIIDR	0xD8C1043B	32-bit	Activity Monitors Implementation Identification Register
0xFA8	AMDEVAFF0	See individual bit resets.	32-bit	Activity Monitors Device Affinity Register 0
0xFAC	AMDEVAFF1	See individual bit resets.	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	0x0000008C	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	0x00000000	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

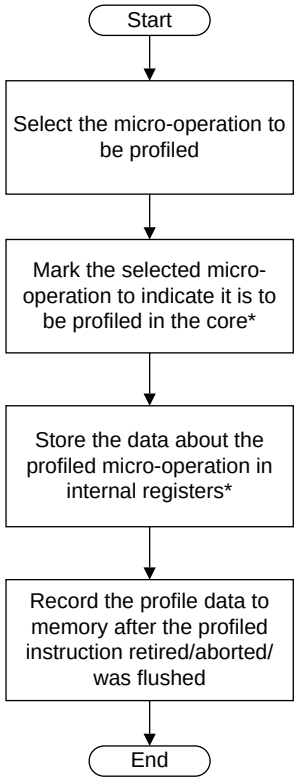
21. Statistical Profiling Extension support

The C1-Ultra core 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-Ultra core profiles micro-operations to minimize the amount of logic necessary to support the SPE.

The following figure shows the SPE behavior in the C1-Ultra core.

Figure 21-1: SPE behavior



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

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

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


Note

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

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

21.1 Statistical Profiling Extension events packet

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

The following table shows the events defined in the 32-bit events packet implemented in the C1-Ultra core.

Table 21-1: SPE events packet

Bits	Definition
[31:26]	Reserved
[25]	Streaming Mode Compute Unit (SMCU) or external coprocessor operation
[24]	Streaming Scalable Vector Extension (SVE) mode
[23]	Reserved
[22]	Recently fetched cache line
[21:19]	Reserved
[18]	Empty predicate
[17]	Partial predicate
[16:13]	Reserved
[12]	Late prefetch
[11]	Data alignment flag
[10]	Remote access
[9]	Last level cache miss
[8]	Last level cache access
[7]	Branch mispredicted
[6]	Not taken
[5]	Translation Lookaside Buffer (TLB) walk
[4]	TLB access
[3]	L1 data cache refill
[2]	L1 data cache access

Bits	Definition
[1]	Architecturally retired
[0]	Generated exception

21.2 Statistical Profiling Extension data source packet

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

The following table shows the data source defined in the 8-bit data source packet implemented in the C1-Ultra core.

Table 21-2: SPE data source packet

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

21.3 AArch64 Statistical Profiling Extension registers

The following summary table provides an overview of all SPE registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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 21-3: SPE registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMSCR_EL1	3	0	C9	C9	0	See individual bit resets.	64-bit	Statistical Profiling Control Register (EL1)
PMSNEVFR_EL1	3	0	C9	C9	1	See individual bit resets.	64-bit	Sampling Inverted Event Filter Register
PMSICR_EL1	3	0	C9	C9	2	See individual bit resets.	64-bit	Sampling Interval Counter Register
PMSIRR_EL1	3	0	C9	C9	3	See individual bit resets.	64-bit	Sampling Interval Reload Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMSFCR_EL1	3	0	C9	C9	4	See individual bit resets.	64-bit	Sampling Filter Control Register
PMSEVFR_EL1	3	0	C9	C9	5	See individual bit resets.	64-bit	Sampling Event Filter Register
PMSLATFR_EL1	3	0	C9	C9	6	See individual bit resets.	64-bit	Sampling Latency Filter Register
PMSIDR_EL1	3	0	C9	C9	7	See individual bit resets.	64-bit	Sampling Profiling ID Register
PMBLIMITR_EL1	3	0	C9	C10	0	See individual bit resets.	64-bit	Profiling Buffer Limit Address Register
PMBPTR_EL1	3	0	C9	C10	1	See individual bit resets.	64-bit	Profiling Buffer Write Pointer Register
PMBSR_EL1	3	0	C9	C10	3	See individual bit resets.	64-bit	Profiling Buffer Status/syndrome Register (EL1)
PMBIDR_EL1	3	0	C9	C10	7	0x0000000000000226	64-bit	Profiling Buffer ID Register
PMSCR_EL2	3	4	C9	C9	0	See individual bit resets.	64-bit	Statistical Profiling Control Register (EL2)

Appendix A AArch64 registers

This appendix contains the descriptions for the C1-Ultra 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 AMU registers summary

The following summary table provides an overview of all AMU registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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: AMU registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMCR_EL0	3	3	C13	C2	0	See individual bit resets.	64-bit	Activity Monitors Control Register
AMCFGR_EL0	3	3	C13	C2	1	0x0000000011003F09	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	3	3	C13	C2	2	0x0000000000000604	64-bit	Activity Monitors Counter Group Configuration Register
AMUSERENR_EL0	3	3	C13	C2	3	See individual bit resets.	64-bit	Activity Monitors User Enable Register
AMCNTENCLR0_EL0	3	3	C13	C2	4	See individual bit resets.	64-bit	Activity Monitors Count Enable Clear Register 0
AMCNTENSET0_EL0	3	3	C13	C2	5	See individual bit resets.	64-bit	Activity Monitors Count Enable Set Register 0
AMCNTENCLR1_EL0	3	3	C13	C3	0	See individual bit resets.	64-bit	Activity Monitors Count Enable Clear Register 1
AMCNTENSET1_EL0	3	3	C13	C3	1	See individual bit resets.	64-bit	Activity Monitors Count Enable Set Register 1
AMEVCNTR00_EL0	3	3	C13	C4	0	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR01_EL0	3	3	C13	C4	1	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR02_EL0	3	3	C13	C4	2	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR03_EL0	3	3	C13	C4	3	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 0
AMEVTYPER00_EL0	3	3	C13	C6	0	0x0000000000000011	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	3	C13	C6	1	0x0000000000004004	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	3	C13	C6	2	0x0000000000000008	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	3	C13	C6	3	0x0000000000004005	64-bit	Activity Monitors Event Type Registers 0
AMEVCNTR10_EL0	3	3	C13	C12	0	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR11_EL0	3	3	C13	C12	1	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR12_EL0	3	3	C13	C12	2	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR13_EL0	3	3	C13	C12	3	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMEVCNTR14_EL0	3	3	C13	C12	4	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR15_EL0	3	3	C13	C12	5	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
AMEVTYPER10_EL0	3	3	C13	C14	0	0x0000000000000300	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	3	C13	C14	1	0x0000000000000301	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	3	C13	C14	2	0x0000000000000302	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER13_EL0	3	3	C13	C14	3	0x0000000000000310	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER14_EL0	3	3	C13	C14	4	0x0000000000003200	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER15_EL0	3	3	C13	C14	5	0x0000000000003202	64-bit	Activity Monitors Event Type Registers 1

A.1.1 AMCFGR_EL0, Activity Monitors Configuration Register

Global configuration register for the activity monitors.

Provides information on supported features, the number of counter groups implemented, the total number of activity monitor event counters implemented, and the size of the counters. AMCFGR_EL0 is applicable to both the architected and the auxiliary counter groups.

Configurations

AArch64 register AMCFGR_EL0 bits [31:0] are architecturally mapped to External register B.1.15 AMCFGR, Activity Monitors Configuration Register on page 641 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0001 0001 0000 0000 0011 1111 0000 1001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure A-1: AARCH64_AMCFGR_EL0 bit assignments

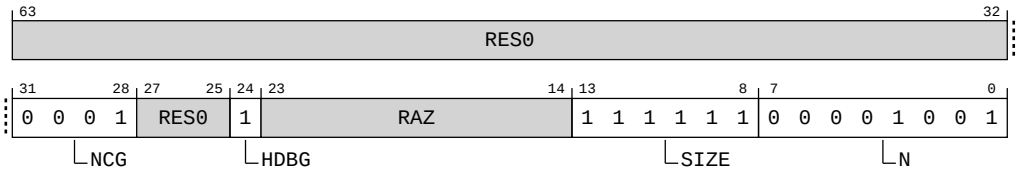


Table A-2: AMCFGR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[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** AMCR_EL0.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. **0x09** Ten activity monitor event counters	0x09

Access

MRS <Xt>, AMCFGR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0010	0b001

Accessibility

MRS <Xt>, AMCFGR_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCFGR_EL0;
elsif PSTATE.EL == EL1 then
```

```
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCFGR_EL0;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCFGR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMCFGR_EL0;
```

A.1.2 AMCGCR_EL0, Activity Monitors Counter Group Configuration Register

Provides information on the number of activity monitor event counters implemented within each counter group.

Configurations

AArch64 register AMCGCR_EL0 bits [31:0] are architecturally mapped to External register B.1.14 AMCGCR, Activity Monitors Counter Group Configuration Register on page 640 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0110 0000 0100
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure A-2: AARCH64_AMCGCR_EL0 bit assignments

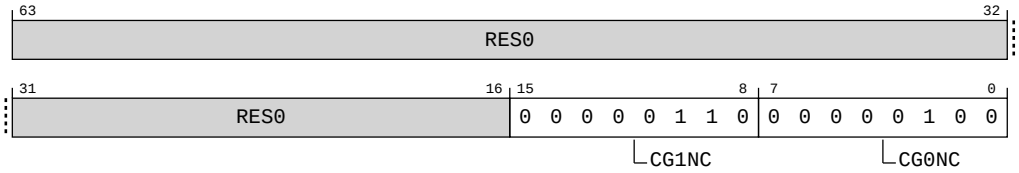


Table A-4: AMCGCR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15:8]	CG1NC	Counter Group 1 Number of Counters. The number of counters in the auxiliary counter group. **0x06** Six counters in the auxiliary counter group	0x06
[7:0]	CG0NC	Counter Group 0 Number of Counters. The number of counters in the architected counter group. **0x04**	0x04

Access

MRS <Xt>, AMCGCR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0010	0b010

Accessibility

MRS <Xt>, AMCGCR_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCGCR_EL0;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCGCR_EL0;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCGCR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMCGCR_EL0;
```

A.1.3 AMEVTYPER00_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMEVCNTR00_EL0 counts.

Configurations

AArch64 register AMEVTYPER00_EL0 bits [31:0] are architecturally mapped to External register B.1.4 AMEVTYPER00, Activity Monitors Event Type Registers 0 on page 627 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

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

Bit descriptions

Figure A-3: AARCH64_AMEVTYPER00_EL0 bit assignments

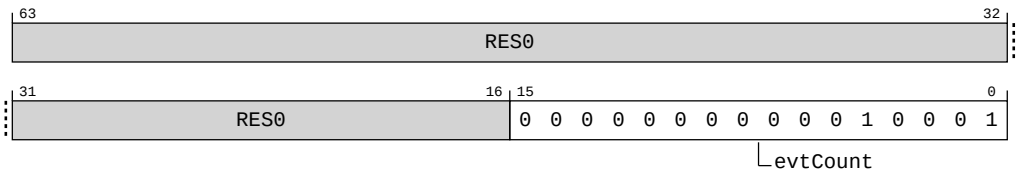


Table A-6: AMEVTYPER00_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**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 AMEVCNTR00_EL0. The value of this field is architecturally mandated for each architected counter. **0x0011** Processor frequency cycles.	0x0011

Access

MRS <Xt>, AMEVTYPER00_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b000

Accessibility

If 0 is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER00_EL0 are **UNDEFINED**.

Note

AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER00_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[0];
elsif PSTATE.EL == EL2 then
```

```
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[0];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER0_EL0[0];
```

A.1.4 AMEVTYPER01_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMEVCNTR01_EL0 counts.

Configurations

AArch64 register AMEVTYPER01_EL0 bits [31:0] are architecturally mapped to External register B.1.5 AMEVTYPER01, Activity Monitors Event Type Registers 0 on page 628 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

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

Bit descriptions

Figure A-4: AARCH64_AMEVTYPER01_EL0 bit assignments

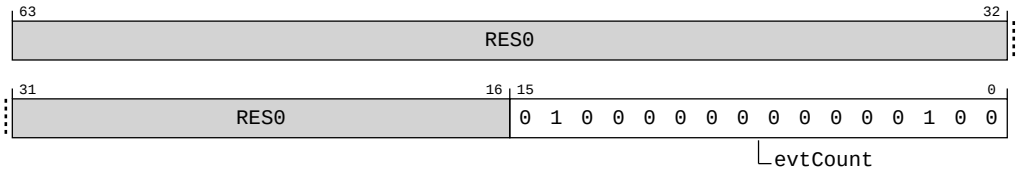


Table A-8: AMEVTYPER01_EL0 bit descriptions

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

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 AMEVCNTR01_EL0. The value of this field is architecturally mandated for each architected counter. **0x4004** Constant frequency cycles.	0x4004

Access

MRS <Xt>, AMEVTYPER01_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b001

Accessibility

If 1 is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER01_EL0 are **UNDEFINED**.

Note

AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER01_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[1];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[1];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER0_EL0[1];
```

A.1.5 AMEVTYPER02_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMEVCNTR02_EL0 counts.

Configurations

AArch64 register AMEVTYPER02_EL0 bits [31:0] are architecturally mapped to External register B.1.6 AMEVTYPER02, Activity Monitors Event Type Registers 0 on page 629 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure A-5: AARCH64_AMEVTYPER02_EL0 bit assignments

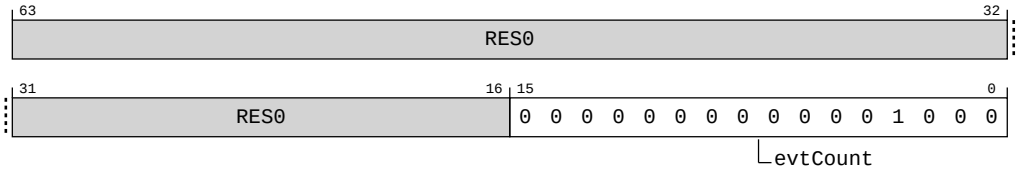


Table A-10: AMEVTYPER02_EL0 bit descriptions

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

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 AMEVCNTR02_EL0. The value of this field is architecturally mandated for each architected counter. **0x0008** Instructions retired.	0x0008

Access

MRS <Xt>, AMEVTYPER02_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b010

Accessibility

If 2 is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER02_EL0 are **UNDEFINED**.

Note

AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER02_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[2];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[2];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER0_EL0[2];
```

A.1.6 AMEVTYPER03_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMEVCNTR03_EL0 counts.

Configurations

AArch64 register AMEVTYPER03_EL0 bits [31:0] are architecturally mapped to External register B.1.7 AMEVTYPER03, Activity Monitors Event Type Registers 0 on page 631 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0100 0000 0000 0101
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-6: AARCH64_AMEVTYPER03_EL0 bit assignments

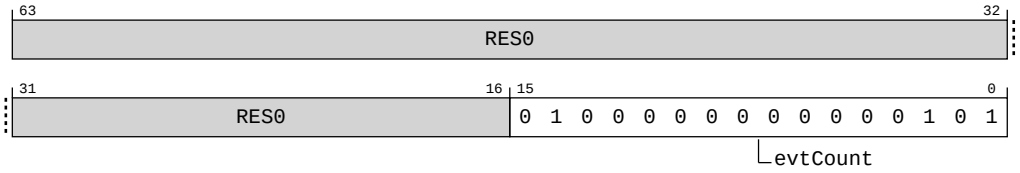


Table A-12: AMEVTYPER03_EL0 bit descriptions

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

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 AMEVCNTR03_EL0. The value of this field is architecturally mandated for each architected counter. **0x4005** Memory stall cycles.	0x4005

Access

MRS <Xt>, AMEVTYPER03_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b011

Accessibility

If 3 is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER03_EL0 are **UNDEFINED**.

Note

AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER03_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[3];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[3];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[3];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER0_EL0[3];
```

A.1.7 AMEVTYPER10_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMEVCNTR10_EL0 counts.

Configurations

AArch64 register AMEVTYPER10_EL0 bits [31:0] are architecturally mapped to External register B.1.8 AMEVTYPER10, Activity Monitors Event Type Registers 1 on page 632 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-7: AARCH64_AMEVTYPER10_EL0 bit assignments

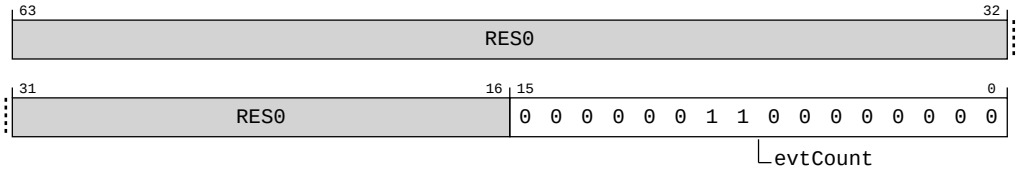


Table A-14: AMEVTYPER10_EL0 bit descriptions

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

Bits	Name	Description	Reset
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AMEVCNTR10_EL0. **0x0300** Gear 0 (MPMM bank 0) period threshold exceeded	0x0300

Access

MRS <Xt>, AMEVTYPER10_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b000

Accessibility

If 0 is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER10_EL0 are **UNDEFINED**.


Note

AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER10_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER10_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVTYPER10_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        X[t, 64] = AMEVTYPER1_EL0[0];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[0];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER1_EL0[0];
```

A.1.8 AMEVTYPER11_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMEVCNTR11_EL0 counts.

Configurations

AArch64 register AMEVTYPER11_EL0 bits [31:0] are architecturally mapped to External register B.1.9 AMEVTYPER11, Activity Monitors Event Type Registers 1 on page 633 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 0000 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-8: AARCH64_AMEVTYPER11_EL0 bit assignments

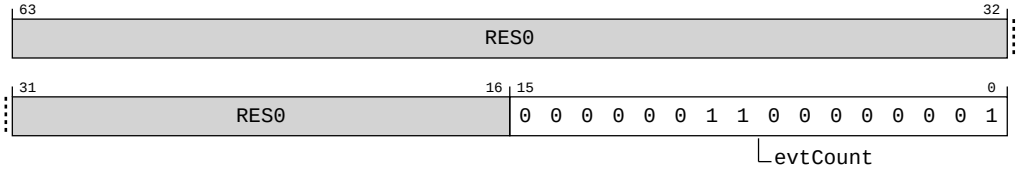


Table A-16: AMEVTYPER11_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**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 AMEVCNTR11_EL0. **0x0301** Gear 1 (MPMM bank 1) period threshold exceeded	0x0301

Access

MRS <Xt>, AMEVTYPER11_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b001

Accessibility

If 1 is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER11_EL0 are **UNDEFINED**.


Note

AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER11_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER11_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVTYPER11_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[1];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[1];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER1_EL0[1];
```

A.1.9 AMEVTYPER12_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMEVCNTR12_EL0 counts.

Configurations

AArch64 register AMEVTYPER12_EL0 bits [31:0] are architecturally mapped to External register B.1.10 AMEVTYPER12, Activity Monitors Event Type Registers 1 on page 635 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 0000 0010
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-9: AARCH64_AMEVTYPER12_EL0 bit assignments

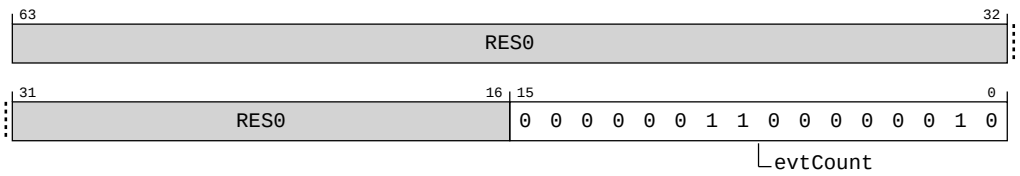


Table A-18: AMEVTYPER12_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**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 AMEVCNTR12_EL0. **0x0302** Gear 2 (MPMM bank 2) period threshold exceeded	0x0302

Access

MRS <Xt>, AMEVTYPER12_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b010

Accessibility

If 2 is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER12_EL0 are **UNDEFINED**.

Note

AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER12_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER12_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVTYPER12_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[2];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[2];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER1_EL0[2];
```

A.1.10 AMEVTYPER13_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMEVCNTR13_EL0 counts.

Configurations

AArch64 register AMEVTYPER13_EL0 bits [31:0] are architecturally mapped to External register B.1.11 AMEVTYPER13, Activity Monitors Event Type Registers 1 on page 636 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 0001 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure A-10: AARCH64_AMEVTYPER13_EL0 bit assignments

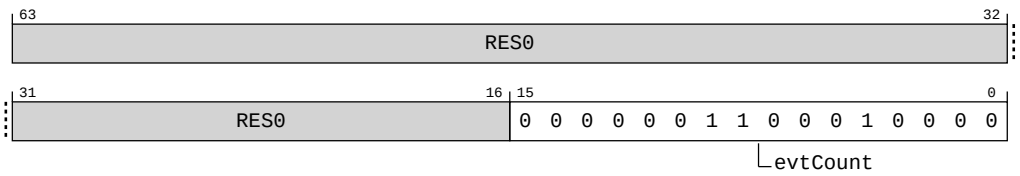


Table A-20: AMEVTYPER13_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**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 AMEVCNTR13_EL0. **0x0310** CPU Activity Count	0x0310

Access

MRS <Xt>, AMEVTYPER13_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b011

Accessibility

If 3 is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER13_EL0 are **UNDEFINED**.

Note

AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER13_EL0

```
if !IsG1ActivityMonitorImplemented(3) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER13_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[3];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVTYPER13_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[3];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[3];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER1_EL0[3];
```

A.1.11 AMEVTYPER14_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMEVCNTR14_EL0 counts.

Configurations

AArch64 register AMEVTYPER14_EL0 bits [31:0] are architecturally mapped to External register AMEVTYPER14, Activity Monitors Event Type Registers 1 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 0010 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure A-11: AARCH64_AMEVTYPER14_EL0 bit assignments

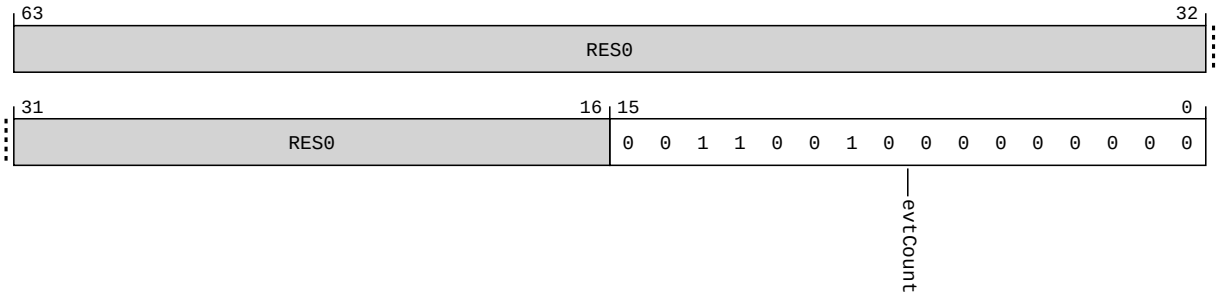


Table A-22: AMEVTYPER14_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**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 AMEVCNTR14_EL0. **0x3200** CPU is stalling because of SME unit backpressure, for any reason	0x3200

Access

```
MRS <Xt>, AMEVTYPER14_EL0
```

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b100

Accessibility

If 4 is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER14_EL0 are **UNDEFINED**.

Note

AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

```
MRS <Xt>, AMEVTYPER14_EL0
```

```
if !IsG1ActivityMonitorImplemented(4) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
```

```
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER14_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[4];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVTYPER14_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[4];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[4];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER1_EL0[4];
```

A.1.12 AMEVTYPER15_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMEVCNTR15_EL0 counts.

Configurations

AArch64 register AMEVTYPER15_EL0 bits [31:0] are architecturally mapped to External register B.1.13 AMEVTYPER15, Activity Monitors Event Type Registers 1 on page 639 bits [31:0].

Attributes

Width

64

Functional group

AMU

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 0010 0000 0010
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-12: AARCH64_AMEVTYPER15_EL0 bit assignments

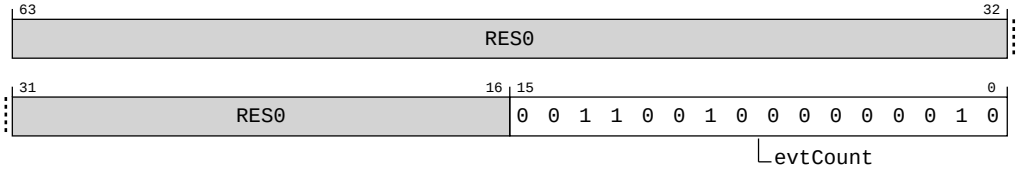


Table A-24: AMEVTYPER15_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**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 AMEVCNTR15_EL0. **0x3202** CPU is stalling because of SME2 unit backpressure, waiting for arbitration	0x3202

Access

`MRS <Xt>, AMEVTYPER15_EL0`

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b101

Accessibility

If 5 is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER15_EL0 are **UNDEFINED**.


Note

AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

`MRS <Xt>, AMEVTYPER15_EL0`

```
if !IsG1ActivityMonitorImplemented(5) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
```

```
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER15_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[5];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVTYPER15_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[5];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[5];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER1_EL0[5];
```

A.2 AArch64 Debug registers summary

The following summary table provides an overview of all Debug registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-26: Debug registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
OSDTRRX_EL1	2	0	C0	C0	2	See individual bit resets.	64-bit	OS Lock Data Transfer Register, Receive

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
DBGBVR0_EL1	2	0	C0	C0	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR0_EL1	2	0	C0	C0	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR0_EL1	2	0	C0	C0	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR0_EL1	2	0	C0	C0	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
DBGBVR1_EL1	2	0	C0	C1	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR1_EL1	2	0	C0	C1	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR1_EL1	2	0	C0	C1	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR1_EL1	2	0	C0	C1	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
MDCCINT_EL1	2	0	C0	C2	0	See individual bit resets.	64-bit	Monitor DCC Interrupt Enable Register
MDSCR_EL1	2	0	C0	C2	2	See individual bit resets.	64-bit	Monitor Debug System Control Register
DBGBVR2_EL1	2	0	C0	C2	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR2_EL1	2	0	C0	C2	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR2_EL1	2	0	C0	C2	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR2_EL1	2	0	C0	C2	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
OSDTRTX_EL1	2	0	C0	C3	2	See individual bit resets.	64-bit	OS Lock Data Transfer Register, Transmit
DBGBVR3_EL1	2	0	C0	C3	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR3_EL1	2	0	C0	C3	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR3_EL1	2	0	C0	C3	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR3_EL1	2	0	C0	C3	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
DBGBVR4_EL1	2	0	C0	C4	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR4_EL1	2	0	C0	C4	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGBVR5_EL1	2	0	C0	C5	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR5_EL1	2	0	C0	C5	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
OSECCR_EL1	2	0	C0	C6	2	See individual bit resets.	64-bit	OS Lock Exception Catch Control Register
MDRAR_EL1	2	0	C1	C0	0	See individual bit resets.	64-bit	Monitor Debug ROM Address Register
OSLAR_EL1	2	0	C1	C0	4	See individual bit resets.	64-bit	OS Lock Access Register
OSLSR_EL1	2	0	C1	C1	4	See individual bit resets.	64-bit	OS Lock Status Register
OSDLR_EL1	2	0	C1	C3	4	See individual bit resets.	64-bit	OS Double Lock Register
DBGPRCR_EL1	2	0	C1	C4	4	See individual bit resets.	64-bit	Debug Power Control Register
DBGCLAIMSET_EL1	2	0	C7	C8	6	See individual bit resets.	64-bit	Debug CLAIM Tag Set Register
DBGCLAIMCLR_EL1	2	0	C7	C9	6	See individual bit resets.	64-bit	Debug CLAIM Tag Clear Register
DBGAUTHSTATUS_EL1	2	0	C7	C14	6	See individual bit resets.	64-bit	Debug Authentication Status Register
MDCCSR_EL0	2	3	C0	C1	0	See individual bit resets.	64-bit	Monitor DCC Status Register
DBGDTR_EL0	2	3	C0	C4	0	See individual bit resets.	64-bit	Debug Data Transfer Register, half-duplex
DBGDTRRX_EL0	2	3	C0	C5	0	See individual bit resets.	64-bit	Debug Data Transfer Register, Receive
DBGDTRTX_EL0	2	3	C0	C5	0	See individual bit resets.	64-bit	Debug Data Transfer Register, Transmit
TRFCR_EL1	3	0	C1	C2	1	See individual bit resets.	64-bit	Trace Filter Control Register (EL1)
MDCR_EL2	3	4	C1	C1	1	See individual bit resets.	64-bit	Monitor Debug Configuration Register (EL2)
TRFCR_EL2	3	4	C1	C2	1	See individual bit resets.	64-bit	Trace Filter Control Register (EL2)
IMP_IDATA0_EL3	3	6	C15	C0	0	0x0000000000000000	64-bit	Instruction Register 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_IDATA1_EL3	3	6	C15	C0	1	0x0000000000000000	64-bit	Instruction Register 1
IMP_IDATA2_EL3	3	6	C15	C0	2	0x0000000000000000	64-bit	Instruction Register 2
IMP_DDATA0_EL3	3	6	C15	C1	0	0x0000000000000000	64-bit	Data Register 0
IMP_DDATA1_EL3	3	6	C15	C1	1	0x0000000000000000	64-bit	Data Register 1
IMP_DDATA2_EL3	3	6	C15	C1	2	0x0000000000000000	64-bit	Data Register 2

A.2.1 IMP_IDATA0_EL3, Instruction Register 0

Contains data from a preceeding RAMINDEX operation

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

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
```

Bit descriptions

Figure A-13: AARCH64_IMP_IDATA0_EL3 bit assignments

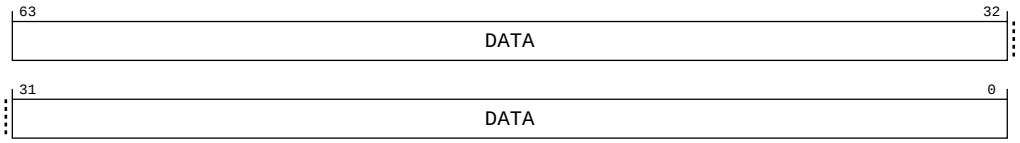


Table A-27: IMP_IDATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	0x0000000000000000

Access

`MRS <Xt>, S3_6_C15_C0_0`

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0000	0b000

Accessibility

`MRS <Xt>, S3_6_C15_C0_0`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_IDATA0_EL3;
```

A.2.2 IMP_IDATA1_EL3, Instruction Register 1

Contains data from a preceeding RAMINDEX operation

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

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
```

Bit descriptions

Figure A-14: AARCH64_IMP_IDATA1_EL3 bit assignments

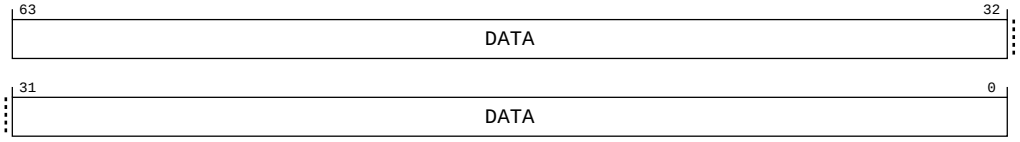


Table A-29: IMP_IDATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	0x0000000000000000

Access

```
MRS <Xt>, S3_6_C15_C0_1
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0000	0b001

Accessibility

```
MRS <Xt>, S3_6_C15_C0_1
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_IDATA1_EL3;
```

A.2.3 IMP_IDATA2_EL3, Instruction Register 2

Contains data from a preceeding RAMINDEX operation

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

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
```

Bit descriptions

Figure A-15: AARCH64_IMP_IDATA2_EL3 bit assignments

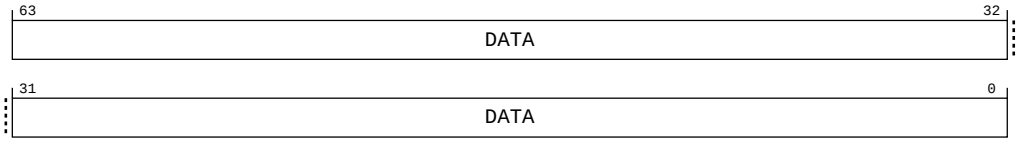


Table A-31: IMP_IDATA2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	0x0000000000000000

Access

`MRS <Xt>, S3_6_C15_C0_2`

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0000	0b010

Accessibility

`MRS <Xt>, S3_6_C15_C0_2`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_IDATA2_EL3;
```

A.2.4 IMP_DDATA0_EL3, Data Register 0

Contains data from a preceeding RAMINDEX operation

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

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
```

Bit descriptions

Figure A-16: AARCH64_IMP_DDATA0_EL3 bit assignments

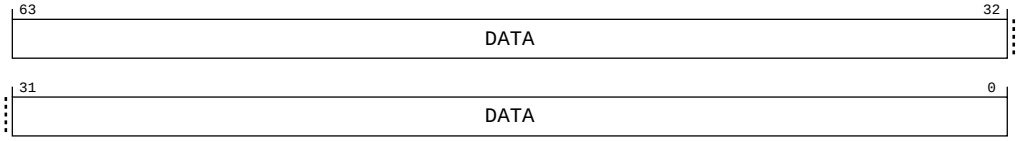


Table A-33: IMP_DDATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	0x0000000000000000

Access

`MRS <Xt>, S3_6_C15_C1_0`

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0001	0b000

Accessibility

`MRS <Xt>, S3_6_C15_C1_0`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_DDATA0_EL3;
```

A.2.5 IMP_DDATA1_EL3, Data Register 1

Contains data from a preceeding RAMINDEX operation

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

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
```

Bit descriptions

Figure A-17: AARCH64_IMP_DDATA1_EL3 bit assignments

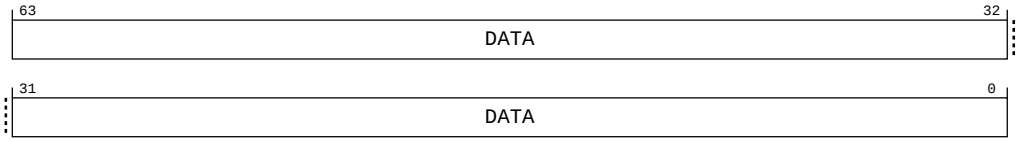


Table A-35: IMP_DDATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	0x0000000000000000

Access

`MRS <Xt>, S3_6_C15_C1_1`

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0001	0b001

Accessibility

`MRS <Xt>, S3_6_C15_C1_1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_DDATA1_EL3;
```

A.2.6 IMP_DDATA2_EL3, Data Register 2

Contains data from a preceeding RAMINDEX operation

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug

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
```

Bit descriptions

Figure A-18: AARCH64_IMP_DDATA2_EL3 bit assignments

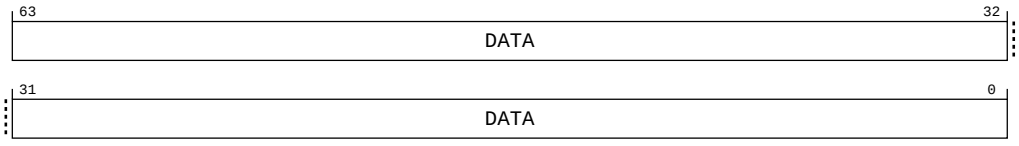


Table A-37: IMP_DDATA2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	0x0000000000000000

Access

`MRS <Xt>, S3_6_C15_C1_2`

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0001	0b010

Accessibility

`MRS <Xt>, S3_6_C15_C1_2`

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

```
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_DDATA2_EL3;
```

A.3 AArch64 GIC system registers summary

The following summary table provides an overview of all GIC system registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-39: GIC system registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICC_PMR_EL1	3	0	C4	C6	0	See individual bit resets.	64-bit	Interrupt Controller Interrupt Priority Mask Register
ICV_PMR_EL1	3	0	C4	C6	0	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Priority Mask Register
ICC_IAR0_EL1	3	0	C12	C8	0	See individual bit resets.	64-bit	Interrupt Controller Interrupt Acknowledge Register 0
ICV_IAR0_EL1	3	0	C12	C8	0	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 0
ICC_EOIR0_EL1	3	0	C12	C8	1	See individual bit resets.	64-bit	Interrupt Controller End Of Interrupt Register 0
ICV_EOIR0_EL1	3	0	C12	C8	1	See individual bit resets.	64-bit	Interrupt Controller Virtual End Of Interrupt Register 0
ICC_HPPIR0_EL1	3	0	C12	C8	2	See individual bit resets.	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 0
ICV_HPPIR0_EL1	3	0	C12	C8	2	See individual bit resets.	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
ICC_BPR0_EL1	3	0	C12	C8	3	See individual bit resets.	64-bit	Interrupt Controller Binary Point Register 0
ICV_BPR0_EL1	3	0	C12	C8	3	See individual bit resets.	64-bit	Interrupt Controller Virtual Binary Point Register 0
ICC_AP0R0_EL1	3	0	C12	C8	4	See individual bit resets.	64-bit	Interrupt Controller Active Priorities Group 0 Registers
ICV_AP0R0_EL1	3	0	C12	C8	4	See individual bit resets.	64-bit	Interrupt Controller Virtual Active Priorities Group 0 Registers
ICC_AP1R0_EL1	3	0	C12	C9	0	See individual bit resets.	64-bit	Interrupt Controller Active Priorities Group 1 Registers
ICV_AP1R0_EL1	3	0	C12	C9	0	See individual bit resets.	64-bit	Interrupt Controller Virtual Active Priorities Group 1 Registers
ICC_NMIAR1_EL1	3	0	C12	C9	5	See individual bit resets.	64-bit	Interrupt Controller Non-maskable Interrupt Acknowledge Register 1

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICV_NMIAR1_EL1	3	0	C12	C9	5	See individual bit resets.	64-bit	Interrupt Controller Virtual Non-maskable Interrupt Acknowledge Register 1
ICC_DIR_EL1	3	0	C12	C11	1	See individual bit resets.	64-bit	Interrupt Controller Deactivate Interrupt Register
ICV_DIR_EL1	3	0	C12	C11	1	See individual bit resets.	64-bit	Interrupt Controller Deactivate Virtual Interrupt Register
ICC_RPR_EL1	3	0	C12	C11	3	See individual bit resets.	64-bit	Interrupt Controller Running Priority Register
ICV_RPR_EL1	3	0	C12	C11	3	See individual bit resets.	64-bit	Interrupt Controller Virtual Running Priority Register
ICC_SGI1R_EL1	3	0	C12	C11	5	See individual bit resets.	64-bit	Interrupt Controller Software Generated Interrupt Group 1 Register
ICC_ASGI1R_EL1	3	0	C12	C11	6	See individual bit resets.	64-bit	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
ICC_SGI0R_EL1	3	0	C12	C11	7	See individual bit resets.	64-bit	Interrupt Controller Software Generated Interrupt Group 0 Register
ICC_IAR1_EL1	3	0	C12	C12	0	See individual bit resets.	64-bit	Interrupt Controller Interrupt Acknowledge Register 1
ICV_IAR1_EL1	3	0	C12	C12	0	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 1
ICC_EOIR1_EL1	3	0	C12	C12	1	See individual bit resets.	64-bit	Interrupt Controller End Of Interrupt Register 1
ICV_EOIR1_EL1	3	0	C12	C12	1	See individual bit resets.	64-bit	Interrupt Controller Virtual End Of Interrupt Register 1
ICC_HPPIR1_EL1	3	0	C12	C12	2	See individual bit resets.	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 1
ICV_HPPIR1_EL1	3	0	C12	C12	2	See individual bit resets.	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
ICC_BPR1_EL1	3	0	C12	C12	3	See individual bit resets.	64-bit	Interrupt Controller Binary Point Register 1
ICV_BPR1_EL1	3	0	C12	C12	3	See individual bit resets.	64-bit	Interrupt Controller Virtual Binary Point Register 1
ICC_CTLR_EL1	3	0	C12	C12	4	See individual bit resets.	64-bit	Interrupt Controller Control Register (EL1)
ICV_CTLR_EL1	3	0	C12	C12	4	See individual bit resets.	64-bit	Interrupt Controller Virtual Control Register
ICC_SRE_EL1	3	0	C12	C12	5	See individual bit resets.	64-bit	Interrupt Controller System Register Enable Register (EL1)
ICC_IGRPEN0_EL1	3	0	C12	C12	6	See individual bit resets.	64-bit	Interrupt Controller Interrupt Group 0 Enable Register
ICV_IGRPEN0_EL1	3	0	C12	C12	6	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Group 0 Enable Register
ICC_IGRPEN1_EL1	3	0	C12	C12	7	See individual bit resets.	64-bit	Interrupt Controller Interrupt Group 1 Enable Register
ICV_IGRPEN1_EL1	3	0	C12	C12	7	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Group 1 Enable Register
ICH_AP0R0_EL2	3	4	C12	C8	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Active Priorities Group 0 Registers
ICH_AP1R0_EL2	3	4	C12	C9	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Active Priorities Group 1 Registers
ICC_SRE_EL2	3	4	C12	C9	5	See individual bit resets.	64-bit	Interrupt Controller System Register Enable Register (EL2)
ICH_HCR_EL2	3	4	C12	C11	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Control Register
ICH_VTR_EL2	3	4	C12	C11	1	0x0000000090280003	64-bit	Interrupt Controller VGIC Type Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICH_MISR_EL2	3	4	C12	C11	2	0x0000000000000000	64-bit	Interrupt Controller Maintenance Interrupt State Register
ICH_EISR_EL2	3	4	C12	C11	3	See individual bit resets.	64-bit	Interrupt Controller End of Interrupt Status Register
ICH_ELRSR_EL2	3	4	C12	C11	5	See individual bit resets.	64-bit	Interrupt Controller Empty List Register Status Register
ICH_VMCR_EL2	3	4	C12	C11	7	See individual bit resets.	64-bit	Interrupt Controller Virtual Machine Control Register
ICH_LR0_EL2	3	4	C12	C12	0	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICH_LR1_EL2	3	4	C12	C12	1	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICH_LR2_EL2	3	4	C12	C12	2	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICH_LR3_EL2	3	4	C12	C12	3	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICC_CTLR_EL3	3	6	C12	C12	4	See individual bit resets.	64-bit	Interrupt Controller Control Register (EL3)
ICC_SRE_EL3	3	6	C12	C12	5	See individual bit resets.	64-bit	Interrupt Controller System Register Enable Register (EL3)
ICC_IGRPEN1_EL3	3	6	C12	C12	7	See individual bit resets.	64-bit	Interrupt Controller Interrupt Group 1 Enable Register (EL3)

A.3.1 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers

Provides information about virtual Group 0 active priorities.

Configurations

This register is present only when IsFeatureImplemented(FEAT_GICv3). Otherwise, direct accesses to ICV_AP0R0_EL1 are UNDEFINED.

Attributes

Width

64

Functional group

GIC

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_ICV_AP0R0_EL1 bit assignments

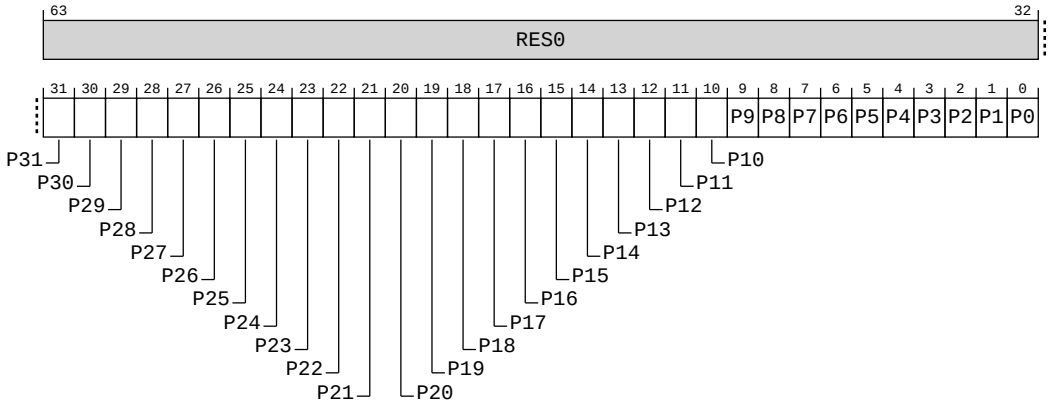


Table A-40: ICV_AP0R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P31	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[30]	P30	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[29]	P29	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[28]	P28	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[27]	P27	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[26]	P26	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[25]	P25	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[24]	P24	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[23]	P23	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[22]	P22	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[21]	P21	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[20]	P20	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[19]	P19	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[18]	P18	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[17]	P17	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[16]	P16	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[15]	P15	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[14]	P14	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[13]	P13	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[12]	P12	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[11]	P11	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[10]	P10	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[9]	P9	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[8]	P8	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[7]	P7	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[6]	P6	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[5]	P5	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[4]	P4	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[3]	P3	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[2]	P2	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[1]	P1	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[0]	P0	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value `0x00000000` is consistent with no interrupts being active.

Access

```
MRS <Xt>, ICC_AP0R0_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b100

```
MSR ICC_AP0R0_EL1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b100

Accessibility

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in **UNPREDICTABLE** behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV_AP0R1_EL1 is implemented only in implementations that support 6 or more bits of priority. ICV_AP0R2_EL1 and ICV_AP0R3_EL1 are implemented only in implementations that support 7 bits of priority. Unimplemented registers are **UNDEFINED**.

Writing to the active priority registers in any order other than the following order might result in **UNPREDICTABLE** behavior of the interrupt prioritization system:

- ICV_AP0R0_EL1.
- ICV_AP1R0_EL1.

`MRS <Xt>, ICC_AP0R0_EL1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_AP0R_EL1[0];
    elsif SCR_EL3.FIQ == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_AP0R_EL1[0];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIQ == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_AP0R_EL1[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_AP0R_EL1[0];
```

```
MSR ICC_AP0R0_EL1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_AP0R_EL1[0] = X[t, 64];
    elsif SCR_EL3.FIQ == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIQ == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R_EL1[0] = X[t, 64];
```

A.3.2 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers

Provides information about virtual Group 1 active priorities.

Configurations

This register is present only when IsFeatureImplemented(FEAT_GICv3). Otherwise, direct accesses to ICV_AP1R0_EL1 are UNDEFINED.

Attributes

Width

64

Functional group

GIC

Access type

RW

Reset value

```
xxxx 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-20: AARCH64_ICV_AP1R0_EL1 bit assignments

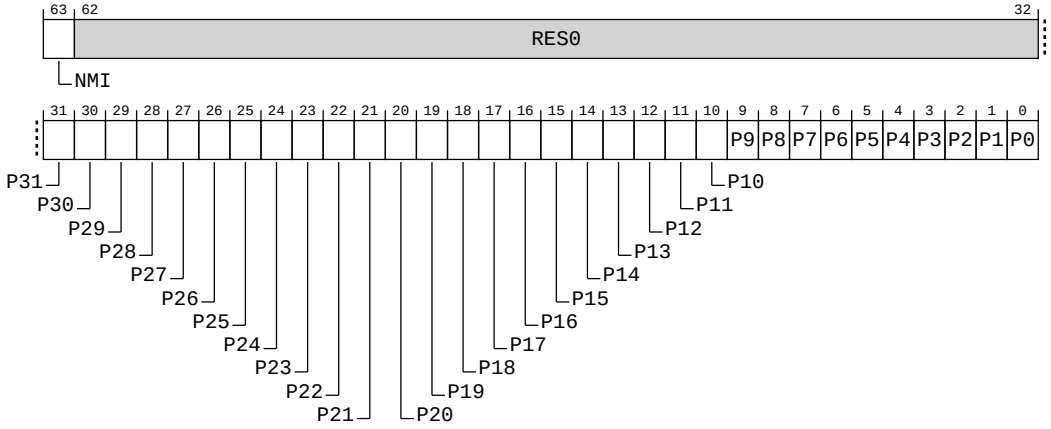


Table A-43: ICV_AP1R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	NMI	**When IsFeatureImplemented(FEAT_GICv3_NMI)** Indicates whether the running priority is from a NMI. **0b0** There is no active Group 1 NMI, or all active Group 1 NMIs have undergone priority-drop. **0b1** There is an active Group 1 NMI. **Otherwise** RES0	0b0
[62:32]	**RES0**	Reserved	**RES0**
[31]	P31	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[30]	P30	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[29]	P29	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[28]	P28	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[27]	P27	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[26]	P26	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[25]	P25	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[24]	P24	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[23]	P23	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[22]	P22	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[21]	P21	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[20]	P20	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[19]	P19	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[18]	P18	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[17]	P17	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[16]	P16	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[15]	P15	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[14]	P14	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[13]	P13	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[12]	P12	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[11]	P11	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[10]	P10	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[9]	P9	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[8]	P8	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[7]	P7	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[6]	P6	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[5]	P5	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[4]	P4	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[3]	P3	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[2]	P2	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[1]	P1	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[0]	P0	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value `0x00000000` is consistent with no interrupts being active.

Access

`MRS <Xt>, ICC_AP1R0_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b000

`MSR ICC_AP1R0_EL1, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b000

Accessibility

Writing to these registers with any value other than the last read value of the register (or `0x00000000` when there are no Group 1 active priorities) might result in **UNPREDICTABLE** behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV_AP1R1_EL1 is implemented only in implementations that support 6 or more bits of priority. ICV_AP1R2_EL1 and ICV_AP1R3_EL1 are implemented only in implementations that support 7 bits of priority. Unimplemented registers are **UNDEFINED**.

Writing to the active priority registers in any order other than the following order might result in **UNPREDICTABLE** behavior of the interrupt prioritization system:

- ICV_AP0R0_EL1.
- ICV_AP1R0_EL1.

`MRS <Xt>, ICC_AP1R0_EL1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.IRQ == '1' then
        UNDEFINED;
```

```
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_AP1R_EL1[0];
    elsif SCR_EL3.IRQ == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[0];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[0];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.IRQ == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[0];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[0];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[0];
```

`MSR ICC_AP1R0_EL1, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_AP1R_EL1[0] = X[t, 64];
    elsif SCR_EL3.IRQ == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R_EL1_S[0] = X[t, 64];
        else
            ICC_AP1R_EL1_NS[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.IRQ == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R_EL1_S[0] = X[t, 64];
        else
            ICC_AP1R_EL1_NS[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R_EL1_S[0] = X[t, 64];
        else
            ICC_AP1R_EL1_NS[0] = X[t, 64];
```

A.3.3 ICC_CTLR_EL1, Interrupt Controller Control Register (EL1)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configurations

This register is present only when IsFeatureImplemented(FEAT_GICv3). Otherwise, direct accesses to ICC_CTLR_EL1 are UNDEFINED.

Attributes

Width

64

Functional group

GIC

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_ICC_CTLR_EL1 bit assignments

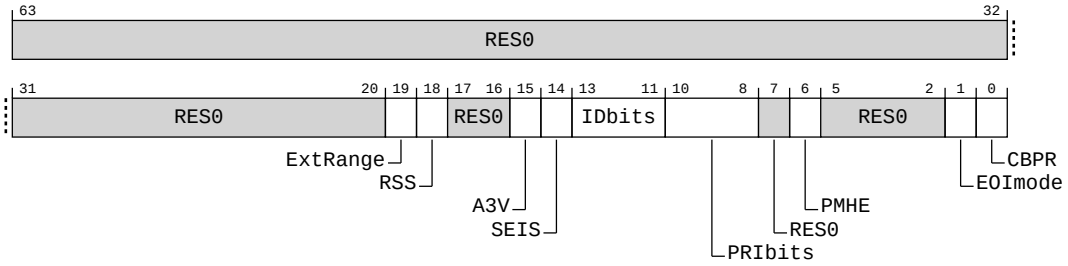


Table A-46: ICC_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	**RES0**	Reserved	**RES0**
[19]	ExtRange	Extended INTID range (read-only). **0b1** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	x
[18]	RSS	Range Selector Support. Possible values are: **0b0** Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	x
[17:16]	**RES0**	Reserved	**RES0**
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. Possible values are: **0b1** The CPU interface logic supports nonzero values of Affinity 3 in SGI generation System registers.	x
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs: **0b0** The CPU interface logic does not support local generation of SEIs.	x
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported: **0b000** 16 bits.	xxx

Bits	Name	Description	Reset
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits). An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits). **Note:** This field always returns the number of priority bits implemented, regardless of the Security state of the access or the value of GICD_CTLR.DS. For physical accesses, this field determines the minimum value of ICC_BPR0_EL1. If EL3 is implemented, physical accesses return the value from ICC_CTLR_EL3.PRIbits. **0b100** Five bits of priority are implemented	xxx
[7]	**RES0**	Reserved	**RES0**
[6]	PMHE	Priority Mask Hint Enable. Controls whether the priority mask register is used as a hint for interrupt distribution: **0b0** Disables use of ICC_PMR_EL1 as a hint for interrupt distribution. **0b1** Enables use of ICC_PMR_EL1 as a hint for interrupt distribution.	x
[5:2]	**RES0**	Reserved	**RES0**
[1]	EOImode	EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt: **0b0** ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality.	x
[0]	CBPR	Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupts: **0b0** ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. ICC_BPR1_EL1 determines the preemption group for Group 1 interrupts. **0b1** ICC_BPR0_EL1 determines the preemption group for both Group 0 and Group 1 interrupts.	x

Access

`MRS <Xt>, ICC_CTLR_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

MSR ICC_CTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

Accessibility

MRS <Xt>, ICC_CTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
```

MSR ICC_CTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
```

```
        ICV_CTLR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
```

A.3.4 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register

Controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

Configurations

This register is present only when IsFeatureImplemented(FEAT_GICv3). Otherwise, direct accesses to ICV_CTLR_EL1 are UNDEFINED.

Attributes

Width

64

Functional group

GIC

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 10xx 1000 0100 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-22: AARCH64_ICV_CTLR_EL1 bit assignments

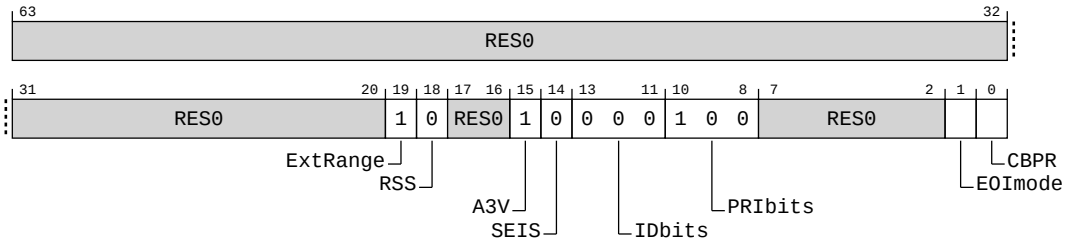


Table A-49: ICV_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	**RES0**	Reserved	**RES0**
[19]	ExtRange	Extended INTID range (read-only). **0b1** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	0b1
[18]	RSS	Range Selector Support. **0b0** Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	0b0
[17:16]	**RES0**	Reserved	**RES0**
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. **0b1** The virtual CPU interface logic supports nonzero values of Affinity 3 in SGI generation System registers.	0b1
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the virtual CPU interface supports local generation of SEIs. **0b0** The virtual CPU interface logic does not support local generation of SEIs.	0b0
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. Indicates the number of virtual interrupt identifier bits supported. **0b000** 16 bits.	0b000

Bits	Name	Description	Reset
[10:8]	PRIbits	Indicates the number of virtual priority bits implemented. An implementation must implement at least 32 levels of virtual priority (5 priority bits). The division between group priority and subpriority is defined in the binary point registers ICV_BPR0_EL1 and ICV_BPR1_EL1. **0b100** Five bits of priority are implemented	0b100
[7:2]	**RES0**	Reserved	**RES0**
[1]	EOImode	Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt: **0b0** ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICV_DIR_EL1 are **UNPREDICTABLE**. **0b1** ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide priority drop functionality only. ICV_DIR_EL1 provides interrupt deactivation functionality.	x
[0]	CBPR	Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts: **0b0** ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts. **0b1** Non-secure reads of ICV_BPR1_EL1 return ICV_BPR0_EL1 plus one, saturated to 0b111. Non-secure writes to ICV_BPR1_EL1 are ignored. Secure reads of ICV_BPR1_EL1 return ICV_BPR0_EL1. Secure writes of ICV_BPR1_EL1 modify ICV_BPR0_EL1.	x

Access

MRS <Xt>, ICC_CTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

MSR ICC_CTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

Accessibility

MRS <Xt>, ICC_CTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
```

`MSR ICC_CTLR_EL1, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
```

```
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
```

A.3.5 ICH_AP0R0_EL2, Interrupt Controller Hyp Active Priorities Group 0 Registers

Provides information about Group 0 virtual active priorities for EL2.

Configurations

If EL2 is not implemented, this register is **RES0** from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

This register is present only when IsFeatureImplemented(FEAT_GICv3). Otherwise, direct accesses to ICH_AP0R0_EL2 are UNDEFINED.

Attributes

Width

64

Functional group

GIC

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 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
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-23: AARCH64_ICH_AP0R0_EL2 bit assignments

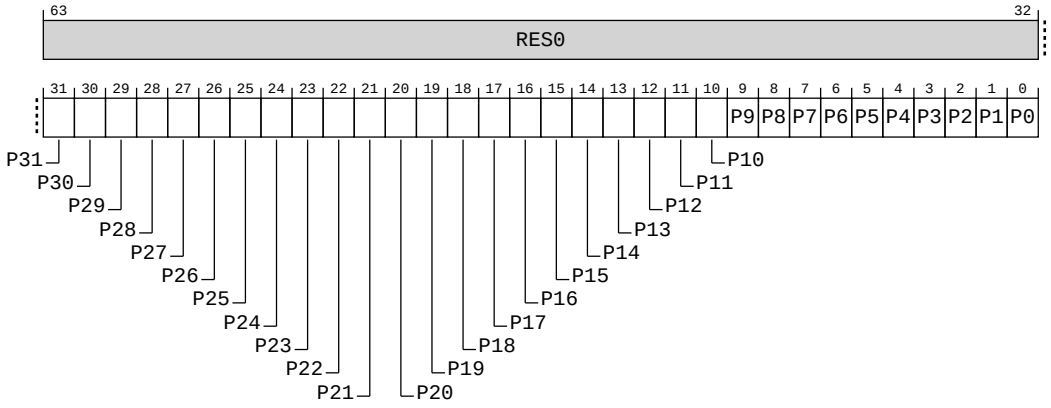


Table A-52: ICH_AP0R0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P31	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[30]	P30	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[29]	P29	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[28]	P28	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0

Bits	Name	Description	Reset
[27]	P27	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[26]	P26	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[25]	P25	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[24]	P24	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[23]	P23	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[22]	P22	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[21]	P21	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0

Bits	Name	Description	Reset
[20]	P20	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[19]	P19	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[18]	P18	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[17]	P17	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[16]	P16	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[15]	P15	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[14]	P14	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0

Bits	Name	Description	Reset
[13]	P13	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[12]	P12	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[11]	P11	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[10]	P10	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[9]	P9	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[8]	P8	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[7]	P7	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0

Bits	Name	Description	Reset
[6]	P6	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[5]	P5	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[4]	P4	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[3]	P3	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[2]	P2	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[1]	P1	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[0]	P0	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0

Software must ensure that ICH_AP0R0_EL2 is 0 for legacy VMs otherwise behavior is **UNPREDICTABLE**. For more information about support for legacy VMs, see 'Support for legacy operation of VMs' in Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (Arm IHI 0069).

The active priorities for Group 0 and Group 1 interrupts for legacy VMs are held in ICH_AP1R0_EL2 and reads and writes to GICV_APR access ICH_AP1R0_EL2. This means that ICH_AP0R0_EL2 is inaccessible to legacy VMs.

Access

```
MRS <Xt>, ICH_AP0R0_EL2
```

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1000	0b000

```
MSR ICH_AP0R0_EL2, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1000	0b000

Accessibility

ICH_AP0R1_EL2 is implemented only in implementations that support 6 or more bits of preemption. ICH_AP0R2_EL2 and ICH_AP0R3_EL2 are implemented only in implementations that support 7 bits of preemption. Unimplemented registers are **UNDEFINED**.


Note

The number of bits of preemption is indicated by ICH_VTR_EL2.PREbits

Writing to these registers with any value other than the last read value of the register (or `0x00000000` for a newly set up virtual machine) can result in **UNPREDICTABLE** behavior of the virtual interrupt prioritization system allowing either:

- Virtual interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution at EL1 or EL0.

Writing to the active priority registers in any order other than the following order will result in **UNPREDICTABLE** behavior:

- ICH_AP0R0_EL2.
- ICH_AP1R0_EL2.

Having the bit corresponding to a priority set in both ICH_AP0R0_EL2 and ICH_AP1R0_EL2 can result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.

```
MRS <Xt>, ICH_AP0R0_EL2
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'1x1'} then
        X[t, 64] = NVMem[0x480 + (8 * 0)];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_AP0R_EL2[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_AP0R_EL2[0];
```

```
MSR ICH_AP0R0_EL2, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'1x1'} then
        NVMem[0x480 + (8 * 0)] = X[t, 64];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_AP0R_EL2[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_AP0R_EL2[0] = X[t, 64];
```

A.3.6 ICH_AP1R0_EL2, Interrupt Controller Hyp Active Priorities Group 1 Registers

Provides information about Group 1 virtual active priorities for EL2.

Configurations

If EL2 is not implemented, this register is **RES0** from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

This register is present only when IsFeatureImplemented(FEAT_GICv3). Otherwise, direct accesses to ICH_AP1R0_EL2 are UNDEFINED.

Attributes

Width

64

Functional group

GIC

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 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
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-24: AARCH64_ICH_AP1R0_EL2 bit assignments

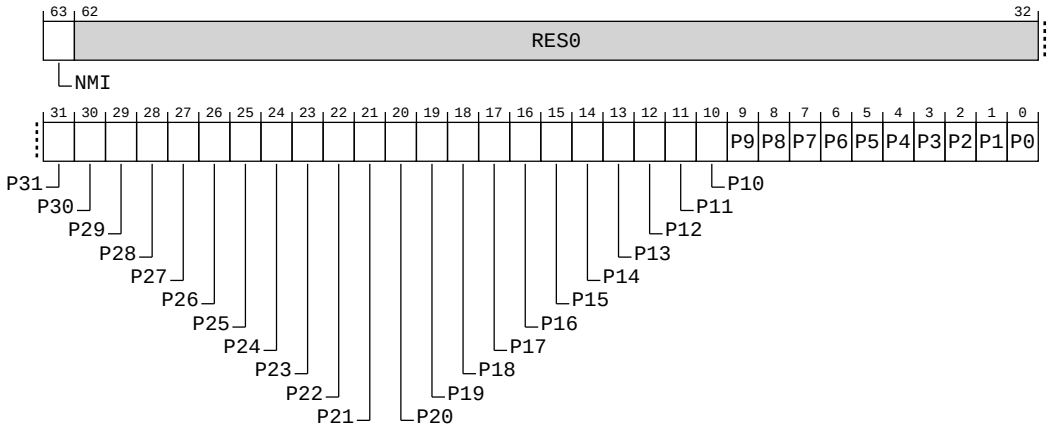


Table A-55: ICH_AP1R0_EL2 bit descriptions

Bits	Name	Description	Reset
[63]	NMI	**When IsFeatureImplemented(FEAT_GICv3_NMI)** Indicates whether the running virtual priority is from a NMI. **0b0** There is no active Group 1 NMI, or all active Group 1 NMIs have undergone priority drop. **0b1** There is an active Group 1 NMI. **Otherwise** RES0	0b0
[62:32]	**RES0**	Reserved	**RES0**
[31]	P31	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[30]	P30	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[29]	P29	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[28]	P28	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[27]	P27	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0

Bits	Name	Description	Reset
[26]	P26	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[25]	P25	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[24]	P24	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[23]	P23	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[22]	P22	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[21]	P21	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[20]	P20	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0

Bits	Name	Description	Reset
[19]	P19	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[18]	P18	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[17]	P17	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[16]	P16	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[15]	P15	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[14]	P14	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[13]	P13	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0

Bits	Name	Description	Reset
[12]	P12	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[11]	P11	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[10]	P10	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[9]	P9	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[8]	P8	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[7]	P7	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[6]	P6	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0

Bits	Name	Description	Reset
[5]	P5	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[4]	P4	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[3]	P3	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[2]	P2	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[1]	P1	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[0]	P0	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0

This register is always used for legacy VMs, regardless of the group of the virtual interrupt. Reads and writes to GICV_APR0 access ICH_AP1R0_EL2. For more information about support for legacy VMs, see 'Support for legacy operation of VMs' in Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (Arm IHI 0069).

Access

```
MRS <Xt>, ICH_AP1R0_EL2
```

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b000

`MSR ICH_AP1R0_EL2, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b000

Accessibility

ICH_AP1R1_EL2 is implemented only in implementations that support 6 or more bits of preemption. ICH_AP1R2_EL2 and ICH_AP1R3_EL2 are implemented only in implementations that support 7 bits of preemption. Unimplemented registers are **UNDEFINED**.


Note

The number of bits of preemption is indicated by ICH_VTR_EL2.PREbits

Writing to these registers with any value other than the last read value of the register (or `0x00000000` for a newly set up virtual machine) can result in **UNPREDICTABLE** behavior of the virtual interrupt prioritization system allowing either:

Writing to the active priority registers in any order other than the following order will result in **UNPREDICTABLE** behavior:

`MRS <Xt>, ICH_AP1R0_EL2`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'1x1'} then
        X[t, 64] = NVMem[0x4A0 + (8 * 0)];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_AP1R_EL2[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_AP1R_EL2[0];
```

`MSR ICH_AP1R0_EL2, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'1x1'} then
        NVMem[0x4A0 + (8 * 0)] = X[t, 64];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_AP1R_EL2[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_AP1R_EL2[0] = X[t, 64];
```

A.3.7 ICH_VTR_EL2, Interrupt Controller VGIC Type Register

Reports supported GIC virtualization features.

Configurations

If EL2 is not implemented, all bits in this register are **RES0** from EL3, except for nV4, which is **RES1** from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

This register is present only when IsFeatureImplemented(FEAT_GICv3). Otherwise, direct accesses to ICH_VTR_EL2 are UNDEFINED.

Attributes

Width

64

Functional group

GIC

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 1001 0000 0010 1000 0000 0000 0000 0011
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-25: AARCH64_ICH_VTR_EL2 bit assignments

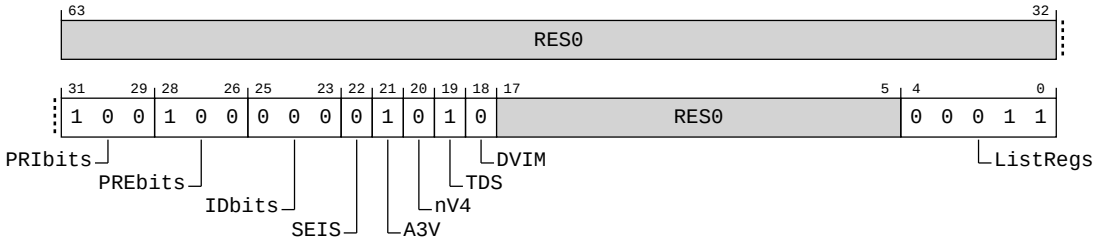


Table A-58: ICH_VTR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:29]	PRIbits	Priority bits. Indicates the number of virtual priority bits implemented, minus one. An implementation must implement at least 32 levels of virtual priority (5 priority bits). This field is an alias of ICV_CTLR_EL1.PRIbits. **0b100** Five virtual priority bits are implemented	0b100
[28:26]	PREbits	Preemption bits. Indicates the number of virtual preemption bits implemented, minus one. An implementation must implement at least 32 levels of virtual preemption priority (5 preemption bits). The value of this field must be less than or equal to the value of ICH_VTR_EL2.PRIbits. This field determines the minimum value of ICH_VMCR_EL2.VBPR0. **0b100** Five virtual preemption bits are implemented	0b100
[25:23]	IDbits	The number of virtual interrupt identifier bits supported: **0b000** 16 bits.	0b000
[22]	SEIS	SEI Support. Indicates whether the virtual CPU interface supports generation of SEIs: **0b0** The virtual CPU interface logic does not support generation of SEIs.	0b0
[21]	A3V	Affinity 3 Valid. **0b1** The virtual CPU interface logic supports nonzero values of Affinity 3 in SGI generation System registers.	0b1
[20]	nV4	Direct injection of virtual interrupts not supported. **0b0** The CPU interface logic supports direct injection of virtual interrupts.	0b0

Bits	Name	Description	Reset
[19]	TDS	Separate trapping of EL1 writes to ICV_DIR_EL1 supported. **0b1** Implementation supports ICH_HCR_EL2.TDIR.	0b1
[18]	DVIM	Masking of directly-injected virtual interrupts. **0b0** Masking of Directly-injected Virtual Interrupts not supported.	0b0
[17:5]	**RES0**	Reserved	**RES0**
[4:0]	ListRegs	List Registers. Indicates the number of List registers implemented, minus one. **0b00011** Four list registers are implemented	0b00011

Access

MRS <Xt>, ICH_VTR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1011	0b001

Accessibility

MRS <Xt>, ICH_VTR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_VTR_EL2;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_VTR_EL2;
```

A.3.8 ICC_CTLR_EL3, Interrupt Controller Control Register (EL3)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configurations

This register is present only when IsFeatureImplemented(FEAT_GICv3). Otherwise, direct accesses to ICC_CTLR_EL3 are UNDEFINED.

Attributes

Width

64

Functional group

GIC

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x0xx 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_ICC_CTLR_EL3 bit assignments

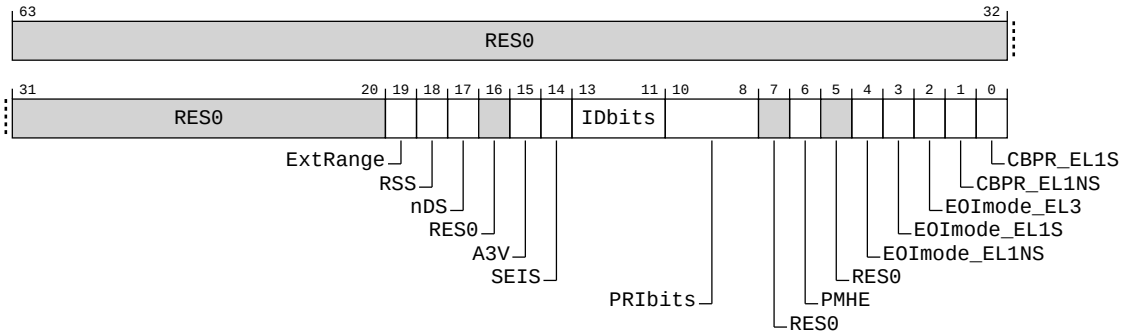


Table A-60: ICC_CTLR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:20]	**RES0**	Reserved	**RES0**
[19]	ExtRange	Extended INTID range (read-only). **0b1** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	x
[18]	RSS	Range Selector Support. **0b0** Targeted SGIs with affinity level 0 values of 0-15 are supported.	x

Bits	Name	Description	Reset
[17]	nDS	Disable Security not supported. Read-only and writes are ignored. **0b1** The CPU interface logic does not support disabling of security, and requires that security is not disabled.	x
[16]	**RES0**	Reserved	**RES0**
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. **0b1** The CPU interface logic supports nonzero values of the Aff3 field in SGI generation System registers.	x
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports generation of SEIs: **0b0** The CPU interface logic does not support generation of SEIs.	x
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. Indicates the number of physical interrupt identifier bits supported. **0b000** 16 bits.	xxx
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits). An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits). **Note:** This field always returns the number of priority bits implemented, regardless of the value of SCR_EL3.NS or the value of GICD_CTLR.DS. The division between group priority and subpriority is defined in the binary point registers ICC_BPR0_EL1 and ICC_BPR1_EL1. This field determines the minimum value of ICC_BPR0_EL1. **0b100** Five bits of priority are implemented	xxx
[7]	**RES0**	Reserved	**RES0**
[6]	PMHE	Priority Mask Hint Enable. **0b0** Disables use of the priority mask register as a hint for interrupt distribution. **0b1** Enables use of the priority mask register as a hint for interrupt distribution.	0b0
[5]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[4]	EOImode_EL1NS	EOI mode for interrupts handled at Non-secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt. **0b0** ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality.	x
[3]	EOImode_EL1S	EOI mode for interrupts handled at Secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt. **0b0** ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality.	x
[2]	EOImode_EL3	EOI mode for interrupts handled at EL3. Controls whether a write to an End of Interrupt register also deactivates the interrupt. **0b0** ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality.	x
[1]	CBPR_EL1NS	Common Binary Point Register, EL1 Non-secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Non-secure interrupts at EL1 and EL2. **0b0** ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. ICC_BPR1_EL1 determines the preemption group for Non-secure Group 1 interrupts. **0b1** ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Non-secure Group 1 interrupts. Non-secure accesses to GICC_BPR and ICC_BPR1_EL1 access the state of ICC_BPR0_EL1.	x
[0]	CBPR_EL1S	Common Binary Point Register, EL1 Secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Secure interrupts at EL1 and EL2. **0b0** ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. ICC_BPR1_EL1 determines the preemption group for Secure Group 1 interrupts. **0b1** ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Secure Group 1 interrupts. Secure EL1 accesses to ICC_BPR1_EL1 access the state of ICC_BPR0_EL1.	x

Access

`MRS <Xt>, ICC_CTLR_EL3`

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

MSR ICC_CTLR_EL3, <Xt>

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

Accessibility

MRS <Xt>, ICC_CTLR_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_CTLR_EL3;
```

MSR ICC_CTLR_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_CTLR_EL3 = X[t, 64];
```

A.4 AArch64 Generic System Control registers summary

The following summary table provides an overview of all 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.

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-63: Generic System Control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ACTLR_EL1	3	0	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL1)
RGSR_EL1	3	0	C1	C0	5	See individual bit resets.	64-bit	Random Allocation Tag Seed Register.
GCR_EL1	3	0	C1	C0	6	See individual bit resets.	64-bit	Tag Control Register.
TTBR0_EL1	3	0	C2	C0	0	See individual bit resets.	64-bit	Translation Table Base Register 0 (EL1)
TTBR1_EL1	3	0	C2	C0	1	See individual bit resets.	64-bit	Translation Table Base Register 1 (EL1)
TCR_EL1	3	0	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL1)
TCR2_EL1	3	0	C2	C0	3	See individual bit resets.	64-bit	Extended Translation Control Register (EL1)
APIAKeyLo_EL1	3	0	C2	C1	0	See individual bit resets.	64-bit	Pointer Authentication Key A for Instruction (bits[63:0])
APIAKeyHi_EL1	3	0	C2	C1	1	See individual bit resets.	64-bit	Pointer Authentication Key A for Instruction (bits[127:64])
APIBKeyLo_EL1	3	0	C2	C1	2	See individual bit resets.	64-bit	Pointer Authentication Key B for Instruction (bits[63:0])
APIBKeyHi_EL1	3	0	C2	C1	3	See individual bit resets.	64-bit	Pointer Authentication Key B for Instruction (bits[127:64])
APDAKeyLo_EL1	3	0	C2	C2	0	See individual bit resets.	64-bit	Pointer Authentication Key A for Data (bits[63:0])
APDAKeyHi_EL1	3	0	C2	C2	1	See individual bit resets.	64-bit	Pointer Authentication Key A for Data (bits[127:64])
APDBKeyLo_EL1	3	0	C2	C2	2	See individual bit resets.	64-bit	Pointer Authentication Key B for Data (bits[63:0])
APDBKeyHi_EL1	3	0	C2	C2	3	See individual bit resets.	64-bit	Pointer Authentication Key B for Data (bits[127:64])
APGAKeyLo_EL1	3	0	C2	C3	0	See individual bit resets.	64-bit	Pointer Authentication Key A for Code (bits[63:0])
APGAKeyHi_EL1	3	0	C2	C3	1	See individual bit resets.	64-bit	Pointer Authentication Key A for Code (bits[127:64])
SPSel	3	0	C4	C2	0	See individual bit resets.	64-bit	Stack Pointer Select
CurrentEL	3	0	C4	C2	2	0x000000000000000C	64-bit	Current Exception Level
PAN	3	0	C4	C2	3	See individual bit resets.	64-bit	Privileged Access Never
UAO	3	0	C4	C2	4	See individual bit resets.	64-bit	User Access Override
ALLINT	3	0	C4	C3	0	See individual bit resets.	64-bit	All Interrupt Mask Bit
AFSR0_EL1	3	0	C5	C1	0	See individual bit resets.	64-bit	Auxiliary Fault Status Register 0 (EL1)
AFSR1_EL1	3	0	C5	C1	1	See individual bit resets.	64-bit	Auxiliary Fault Status Register 1 (EL1)
ESR_EL1	3	0	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL1)
TFSR_EL1	3	0	C5	C6	0	See individual bit resets.	64-bit	Tag Fault Status Register (EL1)
TFSRE0_EL1	3	0	C5	C6	1	See individual bit resets.	64-bit	Tag Fault Status Register (EL0).
FAR_EL1	3	0	C6	C0	0	See individual bit resets.	64-bit	Fault Address Register (EL1)
PAR_EL1	3	0	C7	C4	0	See individual bit resets.	64-bit	Physical Address Register
MAIR_EL1	3	0	C10	C2	0	See individual bit resets.	64-bit	Memory Attribute Indirection Register (EL1)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMAIR_EL1	3	0	C10	C3	0	See individual bit resets.	64-bit	Auxiliary Memory Attribute Indirection Register (EL1)
LORSA_EL1	3	0	C10	C4	0	See individual bit resets.	64-bit	LORegion Start Address (EL1)
LOREA_EL1	3	0	C10	C4	1	See individual bit resets.	64-bit	LORegion End Address (EL1)
LORN_EL1	3	0	C10	C4	2	See individual bit resets.	64-bit	LORegion Number (EL1)
LORC_EL1	3	0	C10	C4	3	See individual bit resets.	64-bit	LORegion Control (EL1)
LORID_EL1	3	0	C10	C4	7	0x0000000000040004	64-bit	LORegionID (EL1)
VBAR_EL1	3	0	C12	C0	0	See individual bit resets.	64-bit	Vector Base Address Register (EL1)
ISR_EL1	3	0	C12	C1	0	See individual bit resets.	64-bit	Interrupt Status Register
CONTEXTIDR_EL1	3	0	C13	C0	1	See individual bit resets.	64-bit	Context ID Register (EL1)
TPIDR_EL1	3	0	C13	C0	4	See individual bit resets.	64-bit	EL1 Software Thread ID Register
SCXTNUM_EL1	3	0	C13	C0	7	See individual bit resets.	64-bit	EL1 Read/Write Software Context Number
IMP_CPUACTLR_EL1	3	0	C15	C1	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register (EL1)
IMP_CPUACTLR2_EL1	3	0	C15	C1	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 2 (EL1)
IMP_CPUACTLR3_EL1	3	0	C15	C1	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 3 (EL1)
IMP_CPUACTLR4_EL1	3	0	C15	C1	3	See individual bit resets.	64-bit	CPU Auxiliary Control Register 4 (EL1)
IMP_CPUECTLR_EL1	3	0	C15	C1	4	See individual bit resets.	64-bit	CPU Extended Control Register
IMP_CPUECTLR2_EL1	3	0	C15	C1	5	See individual bit resets.	64-bit	CPU Extended Control Register 2
IMP_CPUL2DIRTYLNCT_EL1	3	0	C15	C2	5	0x0000000000000000	64-bit	CPU L2 Dirty Line Count Register
IMP_CPUSYNCASSISTCR_EL1	3	0	C15	C2	6	See individual bit resets.	64-bit	CPU Synchronization Assist Configuration Register
IMP_CPUPWRCTLR_EL1	3	0	C15	C2	7	See individual bit resets.	64-bit	CPU Power Control Register
IMP_ATCR_EL1	3	0	C15	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register (EL1)
IMP_CPUACTLR5_EL1	3	0	C15	C8	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register 5 (EL1)
IMP_CPUACTLR6_EL1	3	0	C15	C8	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 6 (EL1)
IMP_CPUACTLR7_EL1	3	0	C15	C8	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 7 (EL1)
IMP_CPUACTLR10_EL1	3	0	C15	C8	3	See individual bit resets.	64-bit	CPU Auxiliary Control Register 10 (EL1)
IMP_CPUACTLR8_EL1	3	0	C15	C8	5	See individual bit resets.	64-bit	CPU Auxiliary Control Register 8 (EL1)
IMP_CPUACTLR9_EL1	3	0	C15	C8	6	See individual bit resets.	64-bit	CPU Auxiliary Control Register 9 (EL1)
IMP_CPUACTLR11_EL1	3	0	C15	C8	7	See individual bit resets.	64-bit	CPU Auxiliary Control Register 11 (EL1)
IMP_CPUACTLR12_EL1	3	0	C15	C13	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register 12 (EL1)
IMP_CPUACTLR13_EL1	3	0	C15	C13	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 13 (EL1)
AIDR_EL1	3	1	C0	C0	7	0x0000000000000000	64-bit	Auxiliary ID Register
RNDR	3	3	C2	C4	0	See individual bit resets.	64-bit	Random Number
RNDRRS	3	3	C2	C4	1	See individual bit resets.	64-bit	Random Number Full Entropy
NZCV	3	3	C4	C2	0	See individual bit resets.	64-bit	Condition Flags
DAIF	3	3	C4	C2	1	See individual bit resets.	64-bit	Interrupt Mask Bits
SVCR	3	3	C4	C2	2	See individual bit resets.	64-bit	Streaming Vector Control Register
DIT	3	3	C4	C2	5	See individual bit resets.	64-bit	Data Independent Timing

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SSBS	3	3	C4	C2	6	See individual bit resets.	64-bit	Speculative Store Bypass Safe
TCO	3	3	C4	C2	7	See individual bit resets.	64-bit	Tag Check Override
FPCR	3	3	C4	C4	0	See individual bit resets.	64-bit	Floating-point Control Register
FPSR	3	3	C4	C4	1	See individual bit resets.	64-bit	Floating-point Status Register
TPIDR_EL0	3	3	C13	C0	2	See individual bit resets.	64-bit	EL0 Read/Write Software Thread ID Register
TPIDRRO_EL0	3	3	C13	C0	3	See individual bit resets.	64-bit	EL0 Read-Only Software Thread ID Register
TPIDR2_EL0	3	3	C13	C0	5	See individual bit resets.	64-bit	EL0 Read/Write Software Thread ID Register 2
SCXTNUM_EL0	3	3	C13	C0	7	See individual bit resets.	64-bit	EL0 Read/Write Software Context Number
ACTLR_EL2	3	4	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL2)
HACR_EL2	3	4	C1	C1	7	See individual bit resets.	64-bit	Hypervisor Auxiliary Control Register
TTBR0_EL2	3	4	C2	C0	0	See individual bit resets.	64-bit	Translation Table Base Register 0 (EL2)
TTBR1_EL2	3	4	C2	C0	1	See individual bit resets.	64-bit	Translation Table Base Register 1 (EL2)
TCR_EL2	3	4	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL2)
TCR2_EL2	3	4	C2	C0	3	See individual bit resets.	64-bit	Extended Translation Control Register (EL2)
VTTBR_EL2	3	4	C2	C1	0	See individual bit resets.	64-bit	Virtualization Translation Table Base Register
VTCR_EL2	3	4	C2	C1	2	See individual bit resets.	64-bit	Virtualization Translation Control Register
VSTTBR_EL2	3	4	C2	C6	0	See individual bit resets.	64-bit	Virtualization Secure Translation Table Base Register
VSTCR_EL2	3	4	C2	C6	2	See individual bit resets.	64-bit	Virtualization Secure Translation Control Register
AFSR0_EL2	3	4	C5	C1	0	See individual bit resets.	64-bit	Auxiliary Fault Status Register 0 (EL2)
AFSR1_EL2	3	4	C5	C1	1	See individual bit resets.	64-bit	Auxiliary Fault Status Register 1 (EL2)
ESR_EL2	3	4	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL2)
TFSR_EL2	3	4	C5	C6	0	See individual bit resets.	64-bit	Tag Fault Status Register (EL2)
FAR_EL2	3	4	C6	C0	0	See individual bit resets.	64-bit	Fault Address Register (EL2)
HPFAR_EL2	3	4	C6	C0	4	See individual bit resets.	64-bit	Hypervisor IPA Fault Address Register
MAIR_EL2	3	4	C10	C2	0	See individual bit resets.	64-bit	Memory Attribute Indirection Register (EL2)
AMAIR_EL2	3	4	C10	C3	0	See individual bit resets.	64-bit	Auxiliary Memory Attribute Indirection Register (EL2)
VBAR_EL2	3	4	C12	C0	0	See individual bit resets.	64-bit	Vector Base Address Register (EL2)
CONTEXTIDR_EL2	3	4	C13	C0	1	See individual bit resets.	64-bit	Context ID Register (EL2)
TPIDR_EL2	3	4	C13	C0	2	See individual bit resets.	64-bit	EL2 Software Thread ID Register
SCXTNUM_EL2	3	4	C13	C0	7	See individual bit resets.	64-bit	EL2 Read/Write Software Context Number
IMP_ATCR_EL2	3	4	C15	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register (EL2)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_AVTCR_EL2	3	4	C15	C7	1	See individual bit resets.	64-bit	CPU Virtualization Auxiliary Translation Control Register (EL2)
ACTLR_EL3	3	6	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL3)
SCR_EL3	3	6	C1	C1	0	See individual bit resets.	64-bit	Secure Configuration Register
CPTR_EL3	3	6	C1	C1	2	See individual bit resets.	64-bit	Architectural Feature Trap Register (EL3)
MDCR_EL3	3	6	C1	C3	1	See individual bit resets.	64-bit	Monitor Debug Configuration Register (EL3)
TTBR0_EL3	3	6	C2	C0	0	See individual bit resets.	64-bit	Translation Table Base Register 0 (EL3)
TCR_EL3	3	6	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL3)
AFSR0_EL3	3	6	C5	C1	0	See individual bit resets.	64-bit	Auxiliary Fault Status Register 0 (EL3)
AFSR1_EL3	3	6	C5	C1	1	See individual bit resets.	64-bit	Auxiliary Fault Status Register 1 (EL3)
ESR_EL3	3	6	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL3)
TFSR_EL3	3	6	C5	C6	0	See individual bit resets.	64-bit	Tag Fault Status Register (EL3)
FAR_EL3	3	6	C6	C0	0	See individual bit resets.	64-bit	Fault Address Register (EL3)
MAIR_EL3	3	6	C10	C2	0	See individual bit resets.	64-bit	Memory Attribute Indirection Register (EL3)
AMAIR_EL3	3	6	C10	C3	0	See individual bit resets.	64-bit	Auxiliary Memory Attribute Indirection Register (EL3)
VBAR_EL3	3	6	C12	C0	0	See individual bit resets.	64-bit	Vector Base Address Register (EL3)
RVBAR_EL3	3	6	C12	C0	1	See individual bit resets.	64-bit	Reset Vector Base Address Register (if EL3 implemented)
RMR_EL3	3	6	C12	C0	2	See individual bit resets.	64-bit	Reset Management Register (EL3)
TPIDR_EL3	3	6	C13	C0	2	See individual bit resets.	64-bit	EL3 Software Thread ID Register
SCXTNUM_EL3	3	6	C13	C0	7	See individual bit resets.	64-bit	EL3 Read/Write Software Context Number
IMP_CPUL2SDIRTYLNCT_EL3	3	6	C15	C2	3	0x0000000000000000	64-bit	CPU L2 Secure Dirty Line Count Register
IMP_CPUACTLR_EL3	3	6	C15	C4	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register (EL3)
IMP_ATCR_EL3	3	6	C15	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register (EL3)
IMP_CPUPSELR_EL3	3	6	C15	C8	0	See individual bit resets.	64-bit	Selected Instruction Private Select Register
IMP_CPUPCR_EL3	3	6	C15	C8	1	See individual bit resets.	64-bit	Selected Instruction Private Control Register
IMP_CPUPOR_EL3	3	6	C15	C8	2	See individual bit resets.	64-bit	Selected Instruction Private Opcode Register
IMP_CPUPMR_EL3	3	6	C15	C8	3	See individual bit resets.	64-bit	Selected Instruction Private Mask Register
IMP_CPUPOR2_EL3	3	6	C15	C8	4	See individual bit resets.	64-bit	Selected Instruction Private Opcode Register 2
IMP_CPUPMR2_EL3	3	6	C15	C8	5	See individual bit resets.	64-bit	Selected Instruction Private Mask Register 2
IMP_CPUPFR_EL3	3	6	C15	C8	6	See individual bit resets.	64-bit	Selected Instruction Private Flag Register

A.4.1 ACTLR_EL1, Auxiliary Control Register (EL1)

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

Note

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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_ACTLR_EL1 bit assignments

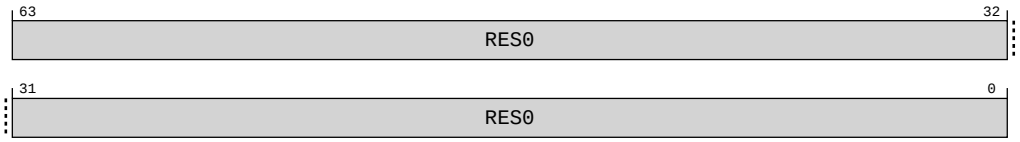


Table A-64: ACTLR_EL1 bit descriptions

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

Access

MRS <Xt>, ACTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b001

MSR ACTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b001

Accessibility

If the **IMPLEMENTATION DEFINED** ACTLR_EL12 accessor is implemented, the following behaviors are also implemented:

- MRS/MSR to ACTLR_EL1 from EL2 accesses ACTLR_EL2.
- MRS/MSR to ACTLR_EL1 from EL1 when the Effective value of HCR_EL2.{NV2, NV1, NV} is {1,0,1} accesses ACTLR_EL1 directly and is not transformed to a memory access.

MRS <Xt>, ACTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ACTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ACTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ACTLR_EL1;
```

MSR ACTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    ACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ACTLR_EL1 = X[t, 64];
```

A.4.2 AFSR0_EL1, Auxiliary Fault Status Register 0 (EL1)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-28: AARCH64_AFSR0_EL1 bit assignments

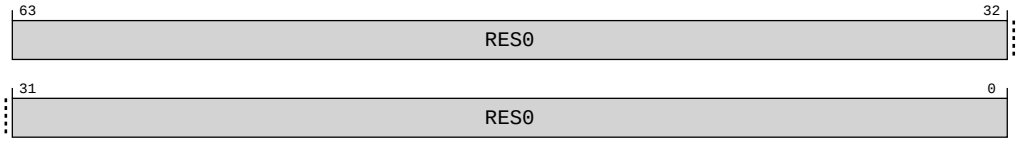


Table A-67: AFSR0_EL1 bit descriptions

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

Access

`MRS <Xt>, AFSR0_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

MSR AFSR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

MRS <Xt>, AFSR0_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b000

MSR AFSR0_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b000

Accessibility

When the Effective value of HCR_EL2.E2H is 1, without explicit synchronization, accesses from EL3 using the accessor name AFSR0_EL1 or AFSR0_EL12 are not guaranteed to be ordered with respect to accesses using the other accessor name.

MRS <Xt>, AFSR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        X[t, 64] = NVMem[0x128];
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        X[t, 64] = AFSR0_EL2;
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR0_EL1;
```

MSR AFSR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        NVMem[0x128] = X[t, 64];
    else
        AFSR0_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        AFSR0_EL2 = X[t, 64];
    else
```

```
        AFSR0_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t, 64];
```

```
MRS <Xt>, AFSR0_EL12
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() == '101' then
        X[t, 64] = NVMem[0x128];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        X[t, 64] = AFSR0_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ELIsInHost(EL2) then
        X[t, 64] = AFSR0_EL1;
    else
        UNDEFINED;
```

```
MSR AFSR0_EL12, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() == '101' then
        NVMem[0x128] = X[t, 64];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        AFSR0_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ELIsInHost(EL2) then
        AFSR0_EL1 = X[t, 64];
    else
        UNDEFINED;
```

A.4.3 AFSR1_EL1, Auxiliary Fault Status Register 1 (EL1)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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_AFSR1_EL1 bit assignments

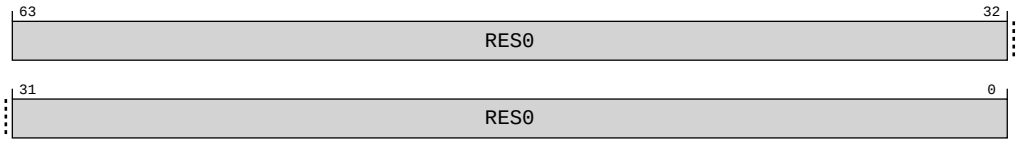


Table A-72: AFSR1_EL1 bit descriptions

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

Access

`MRS <Xt>, AFSR1_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

`MSR AFSR1_EL1, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

`MRS <Xt>, AFSR1_EL12`

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b001

`MSR AFSR1_EL12, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b001

Accessibility

When the Effective value of HCR_EL2.E2H is 1, without explicit synchronization, accesses from EL3 using the accessor name `AFSR1_EL1` or `AFSR1_EL12` are not guaranteed to be ordered with respect to accesses using the other accessor name.

`MRS <Xt>, AFSR1_EL1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        X[t, 64] = NVMem[0x130];
    else
        X[t, 64] = AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        X[t, 64] = AFSR1_EL2;
    else
        X[t, 64] = AFSR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR1_EL1;
```

`MSR AFSR1_EL1, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        NVMem[0x130] = X[t, 64];
    else
        AFSR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        AFSR1_EL2 = X[t, 64];
    else
        AFSR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR1_EL1 = X[t, 64];
```

`MRS <Xt>, AFSR1_EL12`

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

```
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() == '101' then
        X[t, 64] = NVMem[0x130];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        X[t, 64] = AFSR1_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ELIsInHost(EL2) then
        X[t, 64] = AFSR1_EL1;
    else
        UNDEFINED;
```

```
MSR AFSR1_EL12, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() == '101' then
        NVMem[0x130] = X[t, 64];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        AFSR1_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ELIsInHost(EL2) then
        AFSR1_EL1 = X[t, 64];
    else
        UNDEFINED;
```

A.4.4 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register (EL1)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by MAIR_EL1.

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-30: AARCH64_AMAIR_EL1 bit assignments

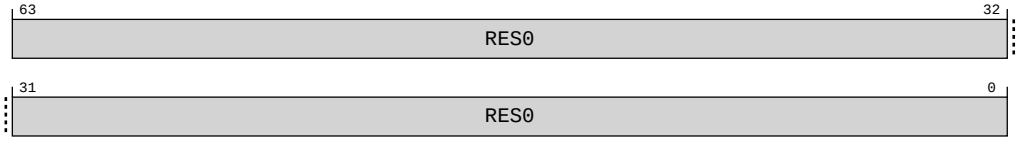


Table A-77: AMAIR_EL1 bit descriptions

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

Access

MRS <Xt>, AMAIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000

MSR AMAIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000

MRS <Xt>, AMAIR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0011	0b000

MSR AMAIR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0011	0b000

Accessibility

When the Effective value of HCR_EL2.E2H is 1, without explicit synchronization, accesses from EL3 using the accessor name `AMAIR_EL1` or `AMAIR_EL12` are not guaranteed to be ordered with respect to accesses using the other accessor name.

`MRS <Xt>, AMAIR_EL1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        X[t, 64] = NVMem[0x148];
    else
        X[t, 64] = AMAIR_EL1;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        X[t, 64] = AMAIR_EL2;
    else
        X[t, 64] = AMAIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR_EL1;
```

`MSR AMAIR_EL1, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        NVMem[0x148] = X[t, 64];
    else
        AMAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        AMAIR_EL2 = X[t, 64];
    else
        AMAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AMAIR_EL1 = X[t, 64];
```

`MRS <Xt>, AMAIR_EL12`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() == '101' then
        X[t, 64] = NVMem[0x148];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        X[t, 64] = AMAIR_EL1;
    else
        UNDEFINED;
```

```
elsif PSTATE.EL == EL3 then
    if ELIsInHost(EL2) then
        X[t, 64] = AMAIR_EL1;
    else
        UNDEFINED;
```

```
MSR AMAIR_EL12, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() == '101' then
        NVMem[0x148] = X[t, 64];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        AMAIR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ELIsInHost(EL2) then
        AMAIR_EL1 = X[t, 64];
    else
        UNDEFINED;
```

A.4.5 LORID_EL1, LORegionID (EL1)

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

Configurations

If no LORegion descriptors are implemented, then the registers LORC_EL1, LORN_EL1, LOREA_EL1, and LORSA_EL1 are **RES0**.

Attributes

Width

64

Functional group

Generic System Control

Access type

RO

Reset value

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

Bit descriptions

Figure A-31: AARCH64_LORID_EL1 bit assignments

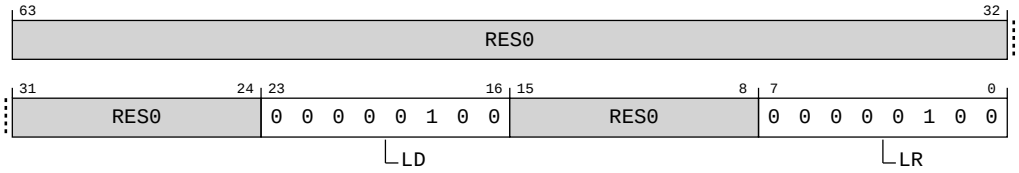


Table A-82: LORID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	**RES0**	Reserved	**RES0**
[23:16]	LD	Number of LORegion descriptors supported by the PE. This is an 8-bit binary number. **0x04** Four LOR descriptors are supported	0x04
[15:8]	**RES0**	Reserved	**RES0**
[7:0]	LR	Number of LORegions supported by the PE. This is an 8-bit binary number. **Note:** If LORID_EL1 indicates that no LORegions are implemented, then LoadLOAcquire and StoreLORelease will behave as LoadAcquire and StoreRelease. **0x04** Four LORegions are supported	0x04

Access

MRS <Xt>, LORID_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b111

Accessibility

MRS <Xt>, LORID_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.LORID_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TLOR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = LORID_EL1;
```

```
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TLOR == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = LORID_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = LORID_EL1;
```

A.4.6 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-32: AARCH64_IMP_CPUACTLR_EL1 bit assignments

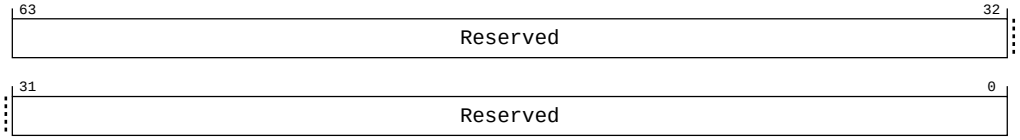


Table A-84: IMP_CPUACTLR_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C1_0

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b000

MSR S3_0_C15_C1_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b000

Accessibility

MRS <Xt>, S3_0_C15_C1_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR_EL1;
```

MSR S3_0_C15_C1_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR_EL1 = X[t, 64];
```

A.4.7 IMP_CPUACTLR2_EL1, CPU Auxiliary Control Register 2 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-33: AARCH64_IMP_CPUACTLR2_EL1 bit assignments

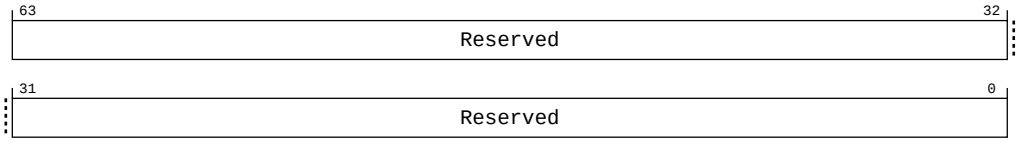


Table A-87: IMP_CPUACTLR2_EL1 bit descriptions

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

Access

```
MRS <Xt>, S3_0_C15_C1_1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b001

```
MSR S3_0_C15_C1_1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b001

Accessibility

```
MRS <Xt>, S3_0_C15_C1_1
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR2_EL1;
```

```
MSR S3_0_C15_C1_1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR2_EL1 = X[t, 64];
```

A.4.8 IMP_CPUACTLR3_EL1, CPU Auxiliary Control Register 3 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-34: AARCH64_IMP_CPUACTLR3_EL1 bit assignments

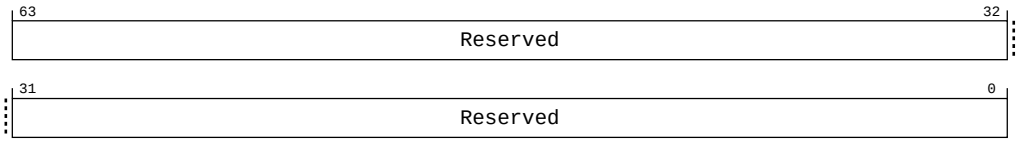


Table A-90: IMP_CPUACTLR3_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C1_2

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b010

MSR S3_0_C15_C1_2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b010

Accessibility

MRS <Xt>, S3_0_C15_C1_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR3_EL1;
elsif PSTATE.EL == EL2 then
```

```
    X[t, 64] = IMP_CPUACTLR3_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR3_EL1;
```

```
MSR S3_0_C15_C1_2, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR3_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR3_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR3_EL1 = X[t, 64];
```

A.4.9 IMP_CPUACTLR4_EL1, CPU Auxiliary Control Register 4 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-35: AARCH64_IMP_CPUACTLR4_EL1 bit assignments

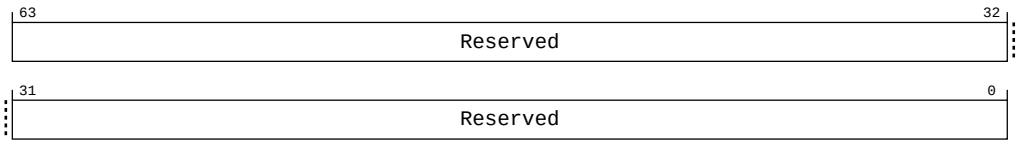


Table A-93: IMP_CPUACTLR4_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C1_3

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b011

MSR S3_0_C15_C1_3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b011

Accessibility

MRS <Xt>, S3_0_C15_C1_3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR4_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR4_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR4_EL1;
```

MSR S3_0_C15_C1_3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR4_EL1 = X[t, 64];
```

```
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR4_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR4_EL1 = X[t, 64];
```

A.4.10 IMP_CPUECTLR_EL1, CPU Extended Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
1000 0000 0000 0xxx 00xx 0100 0000 0x11 0100 0000 0101 01xx 0x11 xx00 xx00 0x00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
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-36: AARCH64_IMP_CPUECTLR_EL1 bit assignments

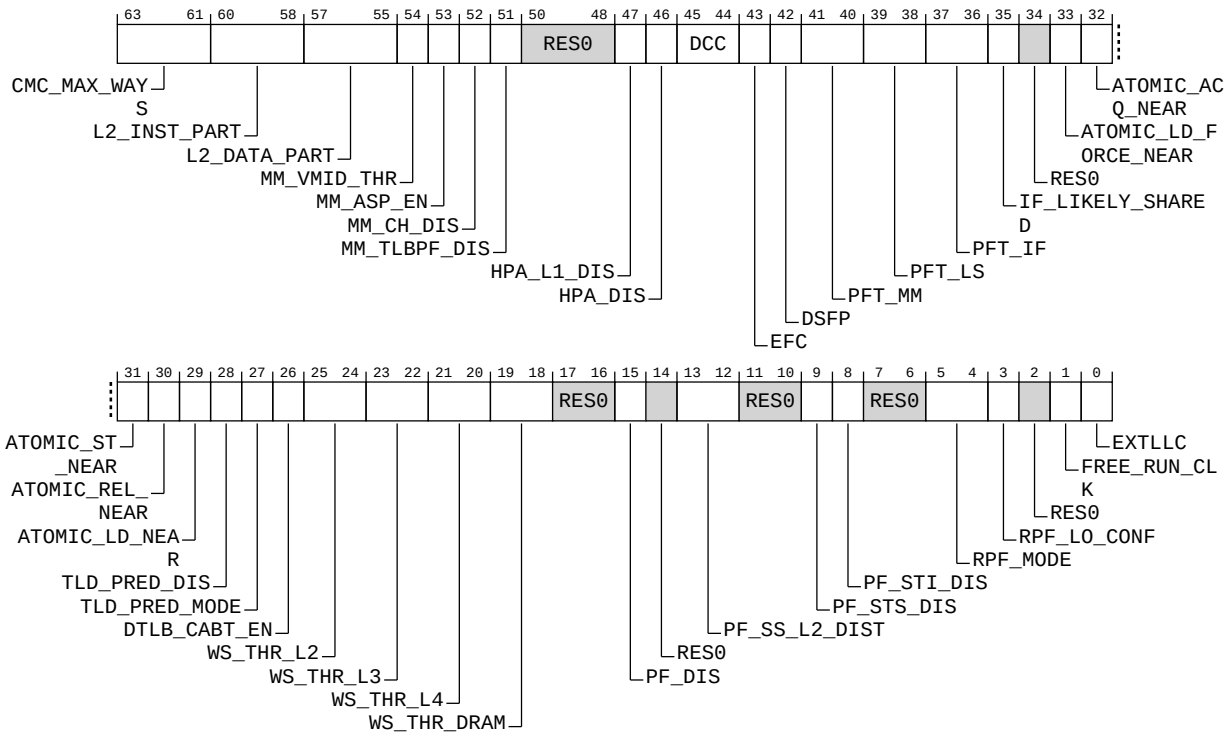


Table A-96: IMP_CPUECTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:61]	CMC_MAX_WAYS	Maximum number of ways of the the L2 used by CMC. The possible values are: **0b000** CMC disabled **0b001** CMC can use 1 way **0b010** CMC can use 2 ways **0b011** CMC can use 3 ways **0b100** CMC can use 4 ways **0b101** CMC can use 5 ways **0b110** CMC can use 6 ways. This is the reset value **0b111** Reserved	0b100

Bits	Name	Description	Reset
[60:58]	L2_INST_PART	Partition the L2 cache for Instruction. The possible values are: **0b000** No ways reserved for instructions. This is the reset value **0b001** Reserve 1 way for instruction. Only instruction fetches can allocate way 7 (2MB) or 11 (3MB) **0b010** Reserve 2 ways for instruction. Only instruction fetches can allocate ways 7:6 (2MB) or 11:10 (3MB) **0b011** Reserve 3 ways for instruction. Only instruction fetches can allocate ways 7:5 (2MB) or 11:9 (3MB) **0b100** Reserve 4 ways for instruction. Only instruction fetches can allocate ways 7:4 (2MB) or 11:8 (3MB) **0b101** Reserve 5 ways for instruction. Only instruction fetches can allocate ways 7:3 (2MB) or 11:7 (3MB) **0b110** Reserve 6 ways for instruction. Only instruction fetches can allocate ways 7:2 (2MB) or 11:6 (3MB) **0b111** Reserve 7 ways for instruction. Only instruction fetches can allocate ways 7:1 (2MB) or 11:5 (3MB)	0b000
[57:55]	L2_DATA_PART	Reserve L2 capacity for data accesses. The possible values are: **0b000** No ways reserved for data. This is the reset value **0b001** Reserve 1 way for data. Only data accesses can allocate way 0 **0b010** Reserve 2 ways for data. Only data accesses can allocate ways 1:0 **0b011** Reserve 3 ways for data. Only data accesses can allocate ways 2:0 **0b100** Reserve 4 ways for data. Only data accesses can allocate ways 3:0 **0b101** Reserve 5 ways for data. Only data accesses can allocate ways 4:0 **0b110** Reserve 6 ways for data. Only data accesses can allocate ways 5:0 **0b111** Reserve 7 ways for data. Only data accesses can allocate ways 6:0	0b000

Bits	Name	Description	Reset
[54]	MM_VMID_THR	VMID filter threshold. The possible values are: **0b0** VMID filter flush after 16 unique VMID allocations to the MMU Translation Cache. This is the default value. **0b1** VMID filter flush after 32 unique VMID allocations to the MMU Translation Cache	0b0
[53]	MM_ASP_EN	Disables allocation of splintered pages in L2 TLB. The possible values are: **0b0** Enables allocation of splintered pages in the L2 TLB. This is the default value. **0b1** Disables allocation of splintered pages in the L2 TLB.	0b0
[52]	MM_CH_DIS	Disables use of contiguous hint. The possible values are: **0b0** Enables use of contiguous hint. This is the default value. **0b1** Disables use of contiguous hint.	0b0
[51]	MM_TLBPF_DIS	Disables TLB prefetcher. The possible values are: **0b0** Enables TLB prefetcher. This is the default value. **0b1** Disables TLB prefetcher.	0b0
[50:48]	**RES0**	Reserved	**RES0**
[47]	HPA_L1_DIS	Disables hardware page aggregation in L1 TLBs. The possible values are: **0b0** Enables hardware page aggregation in L1 TLBs. This is the default value. **0b1** Disables hardware page aggregation in L1 TLBs.	0b0
[46]	HPA_DIS	Disable Hardware page aggregation. The possible values are: **0b0** Enables hardware page aggregation. This is the default value. **0b1** Disables hardware page aggregation.	0b0

Bits	Name	Description	Reset
[45:44]	DCC	Controls whether evictions of clean cache-lines send data on the CHI interface. Set this based on whether there is a cache on the path to memory. The possible values are: **0b00** Disables sending data when clean cache-lines are evicted. **0b01** Enables sending WriteEvictFull transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data. **0b10** Enables sending WriteEvictOrEvict transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data. **0b11** Enables sending WriteEvictOrEvict transactions when Unique Clean or Shared Clean cache-lines are evicted. This is the default value.	xx
[43]	EFC	Eviction Flush Control (EFC). Controls whether hardware cache flushes and DC CISW instruction send data when evicting clan and dirty cache line. If it is known that data is likely to be used soon by another core, setting this bit can improve system performance. The possible values are: **0b0** Disables cache allocation in downstream caches when hardware cache flushes or DC CISW instructions evict a clean or cache line. Downstream Snoop Filter Present (DFSP) controls the sending of Evict transactions **0b1** Enables cache allocation in downstream caches when hardware cache flushes or DC CISW instructions evict a clean or dirty cache line. Downstream Cache Control (DCC) controls the sending of data. DSFP controls the sending of Evict transactions.	0b0
[42]	DSFP	Downstream Snoop Filter Present (DSFP). Enables sending Evict transactions on the CHI interface when clean lines are evicted without data. You must enable this if there is at least one snoop filter in the path to memory **0b0** Disables sending Evict transactions when clean cachelines are evicted without data **0b1** Enables sending of Evict transaction when clean cachelines are evicted without data.	0b1

Bits	Name	Description	Reset
[41:40]	PFT_MM	DRAM prefetch using PrefetchTgt transactions for tablewalk requests. The possible values are: **0b00** Disable prefetchtgt generation for requests from the Memory Management unit (MMU). This is the default value. **0b01** Conservatively generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable. **0b10** Agressively generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable. **0b11** Always generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable.	0b00
[39:38]	PFT_LS	DRAM prefetch using PrefetchTgt transactions for load and store requests. The possible values are: **0b00** Disable prefetchtgt generation for requests from the Load-Store unit (LS). This is the default value. **0b01** Conservatively generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable. **0b10** Agressively generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable. **0b11** Always generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable.	0b00
[37:36]	PFT_IF	DRAM prefetch using PrefetchTgt transactions for instruction fetch requests. The possible values are: **0b00** Disable prefetchtgt generation for requests from the Instruction Fetch unit (IF). This is the default value. **0b01** Conservatively generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable. **0b10** Agressively generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable. **0b11** Always generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable.	0b00

Bits	Name	Description	Reset
[35]	IF_LIKELY_SHARED	Instruction fetch Shared state control. The possible values are: **0b0** Instruction fetch requests do not assert TXREQ LikelyShared. This is the reset value. **0b1** Instruction fetch requests assert TXREQ LikelyShared and request a SharedClean copy of data.	0b0
[34]	**RES0**	Reserved	**RES0**
[33]	ATOMIC_LD_FORCE_NEAR	A load atomic (including SWP & CAS) instruction to WB memory will be performed near. The possible values are: **0b0** Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request. **0b1** Load-atomic will be performed near by bringing the line into the L1D Cache. This is the default value.	0b1
[32]	ATOMIC_ACQ_NEAR	An atomic instruction to WB memory with acquire semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **0b0** Acquire-atomic is near if cache line is already Exclusive, otherwise make far atomic request. **0b1** Acquire-atomic will make up to 1 fill request to perform near. This is the default value.	0b1
[31]	ATOMIC_ST_NEAR	A store atomic instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **0b0** Store-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the default value. **0b1** Store-atomic will make up to 1 fill request to perform near.	0b0
[30]	ATOMIC_REL_NEAR	An atomic instruction to WB memory with release semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **0b0** Release-atomic is near if cache line is already Exclusive, otherwise make far atomic request. **0b1** Release-atomic will make up to 1 fill request to perform near. This is the default value.	0b1
[29]	ATOMIC_LD_NEAR	A load atomic (including SWP & CAS) instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **0b0** Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the default value. **0b1** Load-atomic will make up to 1 fill request to perform near.	0b0

Bits	Name	Description	Reset
[28]	TLD_PRED_DIS	Disable Transient Load Prediction. The possible values are: **0b0** Enables transient load prediction. This is the default value. **0b1** Disables transient load prediction.	0b0
[27]	TLD_PRED_MODE	Aggressive Transient Load Prediction. The possible values are: **0b0** Disables aggressive transient load prediction. This is the default value. **0b1** Enables aggressive transient load prediction.	0b0
[26]	DTLB_CABT_EN	Enables TLB Conflict Data Abort Exception. The possible values are: **0b0** Disables TLB conflict data abort exception. This is the default value. **0b1** Enables TLB conflict data abort exception.	0b0
[25:24]	WS_THR_L2	Threshold for direct stream to L2 cache on store. The possible values are: **0b00** 256B - This is the default value **0b01** 4KB **0b10** 8KB **0b11** Disables direct stream to L2 cache on store.	0b00
[23:22]	WS_THR_L3	Threshold for direct stream to L3 cache on store. The possible values are: **0b00** 128KB **0b01** 256KB - This is the default value **0b10** 512KB **0b11** Disables direct stream to L3 cache on store.	0b01
[21:20]	WS_THR_L4	Threshold for direct stream to L4 cache on store. The possible values are: **0b00** 256KB **0b01** 512KB - This is the default value **0b10** 1MB **0b11** Disables direct stream to L4 cache on store.	0b01

Bits	Name	Description	Reset
[19:18]	WS_THR_DRAM	Threshold for direct stream to DRAM on store. The possible values are: **0b00** 512KB **0b01** 1MB - This is the default value **0b10** 2MB **0b11** Disables direct stream to DRAM on store.	0b01
[17:16]	**RES0**	Reserved	**RES0**
[15]	PF_DIS	Disables hardware prefetching. The possible values are: **0b0** Enables hardware prefetching. This is the default value. **0b1** Disables hardware prefetching.	0b0
[14]	**RES0**	Reserved	**RES0**
[13:12]	PF_SS_L2_DIST	Single cache line stride prefetching L2 distance. The possible values are: **0b00** 22 lines ahead **0b01** 40 lines ahead **0b10** 60 lines ahead **0b11** Dynamic. This is the default value.	0b11
[11:10]	**RES0**	Reserved	**RES0**
[9]	PF_STS_DIS	Disable store-stride prefetches. The possible values are: **0b0** Enables store prefetching. This is the default value. **0b1** Disables store prefetching.	0b0
[8]	PF_STI_DIS	Disable store prefetches at issue (not overridden by ls_hw_pref_disable). The possible values are: **0b0** Enables store prefetching. This is the default value. **0b1** Disable store prefetching.	0b0
[7:6]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[5:4]	RPF_MODE	Region prefetcher aggressiveness. The possible values are: **0b00** Dynamic region prefetch aggressiveness. This is the default value. **0b01** Conservative region prefetching. **0b10** Very Conservative region prefetching. **0b11** Most Conservative region prefetching. This will disable the region prefetcher.	0b00
[3]	RPF_LO_CONF	Region Prefetcher single accesses training behavior. The possible values are: **0b0** Mostly don't train PHT on single access. This is the default value. **0b1** Always train the PHT on single access. This results in fewer prefetch requests.	0b0
[2]	**RES0**	Reserved	**RES0**
[1]	FREE_RUN_CLK	Free-running SRAM clock. The possible values are: **0b0** Normal clock gate behavior. This is the default value. **0b1** Enable clock gate override on all the single-cycle RAMs	0b0
[0]	EXTLLC	Internal or external Last-level cache (LLC) in the system. The possible values are: **0b0** Indicates that an internal Last-level cache is present in the system, and that the DataSource field on the requester CHI interface indicates when data is returned from the LLC. This is used to control how the LL_CACHE* PMU events count. This is the default value. **0b1** Indicates that an external Last-level cache is present in the system, and that the DataSource field on the requester CHI interface indicates when data is returned from the LLC. This is used to control how the LL_CACHE* PMU events count.	0b0

Access

```
MRS <Xt>, S3_0_C15_C1_4
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b100

```
MSR S3_0_C15_C1_4, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b100

Accessibility

`MRS <Xt>, S3_0_C15_C1_4`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUECTLR_EL1;
```

`MSR S3_0_C15_C1_4, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUECTLR_EL1 = X[t, 64];
```

A.4.11 IMP_CPUECTLR2_EL1, CPU Extended Control Register 2

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x010 0001 1000 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 A-37: AARCH64_IMP_CPUECTLR2_EL1 bit assignments

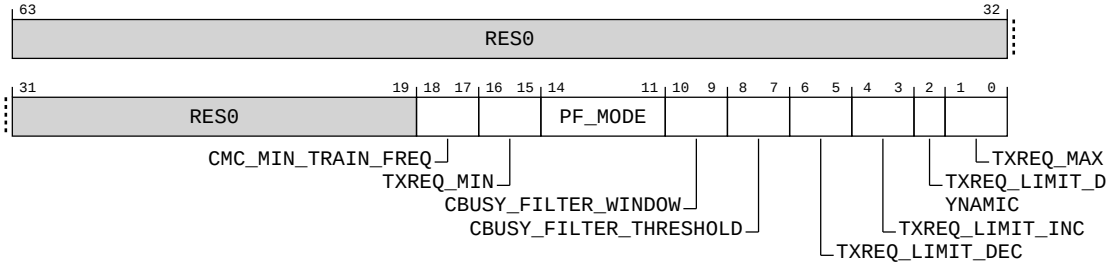


Table A-99: IMP_CPUECTLR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18:17]	CMC_MIN_TRAIN_FREQ	CMC training frequency when in its least aggressive mode. Enables CMC to utilize less than 1 entire way of the L2. The possible values are: **0b00** allow all training in least aggressive mode **0b01** allow 1/8 trains in least aggressive mode - This is the default value **0b10** allow 1/16 trains in least aggressive mode **0b11** allow 1/32 trains in least aggressive mode	0b01
[16:15]	TXREQ_MIN	Minimum number of TXREQ transactions outstanding from the L2 Transaction Queue. The possible values are: **0b00** 1/4 of L2 TQ size - This is the default value **0b01** 1/8 of L2 TQ size **0b10** 1/16 of L2 TQ size **0b11** 1/32 of L2 TQ size	0b00

Bits	Name	Description	Reset
[14:11]	PF_MODE	Prefetcher Aggressiveness Modes. With mode 0 representing the most aggressive mode and 3 representing the most conservative mode. The possible values and associated ranges are: **0b0000** Modes [0,0] (statically at the most aggressive mode) **0b0001** Modes [0,1] **0b0010** Modes [0,2] **0b0011** Modes [0,3] - This is the default value. **0b0100** Modes [1,1] **0b0101** Modes [1,2] **0b0110** Modes [1,3] **0b0111** Modes [2,2] **0b1000** Modes [2,3] **0b1001** Modes [3,3] (statically at the most conservative mode) **0b1010** reserved **0b1011** reserved **0b1100** reserved **0b1101** reserved **0b1110** reserved **0b1111** reserved	0b0011

Bits	Name	Description	Reset
[10:9]	CBUSY_FILTER_WINDOW	Number of CBusy responses in one sampling window. The possible values are: **0b00** 256 - This is the default value **0b01** 64 **0b10** 128 **0b11** 512	0b00
[8:7]	CBUSY_FILTER_THRESHOLD	Fraction of of CBusy responses in the sampling window necessary to be considered a valid sample of that CBusy value. The possible values are: **0b00** 1/16 - This is the default value **0b01** 1/32 **0b10** 1/8 **0b11** 1/4	0b00
[6:5]	TXREQ_LIMIT_DEC	Dynamic TXREQ limit decrement. Controls how quickly the dynamic TXREQ limit is decreased when CBusy indicates value of 3. The possible values are: **0b00** 4 - This is the default value **0b01** 8 **0b10** 16 **0b11** 2	0b00
[4:3]	TXREQ_LIMIT_INC	Dynamic TXREQ limit increment. Controls how quickly the dynamic TXREQ limit is increased when CBusy indicates values less than 2. The possible values are: **0b00** 4 - This is the default value **0b01** 8 **0b10** 16 **0b11** 2	0b00

Bits	Name	Description	Reset
[2]	TXREQ_LIMIT_DYNAMIC	Selects static or dynamic control of TXREQ limit. Dynamic TXREQ limit will adjust based on CBusy responses on RXDAT and RXRSP in the range of the static limit selected by CPUECTLR2_EL1[1:0] and 1/4 of the L2 TQ SIZE. The possible values are: **0b0** maximum number of TXREQ transactions statically set by CPUECTLR2_EL1[1:0] - This is the default value. **0b1** maximum number of TXREQ transactions dynamically controlled	0b0
[1:0]	TXREQ_MAX	Maximum number of TXREQ transactions outstanding from the L2 Transaction Queue. The possible values are: **0b00** full L2 TQ size - This is the default value **0b01** 3/4 of L2 TQ size **0b10** 1/2 of L2 TQ size **0b11** 1/4 of L2 TQ size	0b00

Access

MRS <Xt>, S3_0_C15_C1_5

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b101

MSR S3_0_C15_C1_5, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b101

Accessibility

MRS <Xt>, S3_0_C15_C1_5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUECTLR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUECTLR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUECTLR2_EL1;
```

MSR S3_0_C15_C1_5, <Xt>

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

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUECTLR2_EL1 = X[t, 64];
```

A.4.12 IMP_CPUL2DIRTYLNCT_EL1, CPU L2 Dirty Line Count Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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
```

Bit descriptions

Figure A-38: AARCH64_IMP_CPUL2DIRTYLNCT_EL1 bit assignments

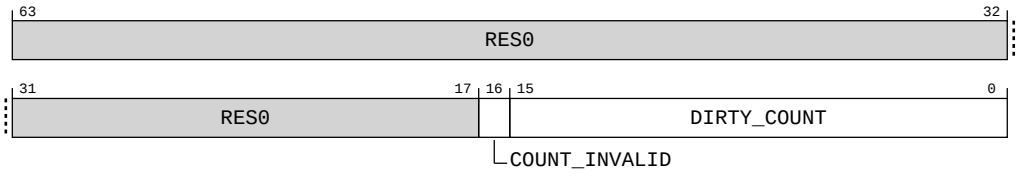


Table A-102: IMP_CPUL2DIRTYLNCT_EL1 bit descriptions

Bits	Name	Description	Reset
[63:17]	**RES0**	Reserved	**RES0**
[16]	COUNT_INVALID	Indicates the dirty count is invalid. Reset value is 'b0	0b0
[15:0]	DIRTY_COUNT	Number of dirty lines in the L2. Reset value is 'h0000	0x0000

Access

`MRS <Xt>, S3_0_C15_C2_5`

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b101

Accessibility

`MRS <Xt>, S3_0_C15_C2_5`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.L2DIRTYEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.L2DIRTYEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = IMP_CPUL2DIRTYLNCT_EL1;
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.L2DIRTYEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = IMP_CPUL2DIRTYLNCT_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUL2DIRTYLNCT_EL1;
```

A.4.13 IMP_CPUSYNCASSISTCR_EL1, CPU Synchronization Assist Configuration Register

This register controls the synchronization assists deferring wakeup for didt events

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-39: AARCH64_IMP_CPUSYNCASSISTCR_EL1 bit assignments

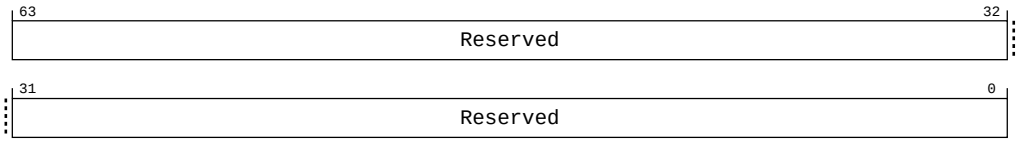


Table A-104: IMP_CPUSYNCASSISTCR_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C2_6

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b110

MSR S3_0_C15_C2_6, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b110

Accessibility

MRS <Xt>, S3_0_C15_C2_6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUSYNCASSISTCR_EL1;
elsif PSTATE.EL == EL2 then
```

```
    X[t, 64] = IMP_CPUSYNCASSISTCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUSYNCASSISTCR_EL1;
```

```
MSR S3_0_C15_C2_6, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ASSISTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ASSISTEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUSYNCASSISTCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ASSISTEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUSYNCASSISTCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUSYNCASSISTCR_EL1 = X[t, 64];
```

A.4.14 IMP_CPUPWRCTLR_EL1, CPU Power Control Register

This register controls various power aspects of the core

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 0000 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-40: AARCH64_IMP_CPUPWRCTLR_EL1 bit assignments

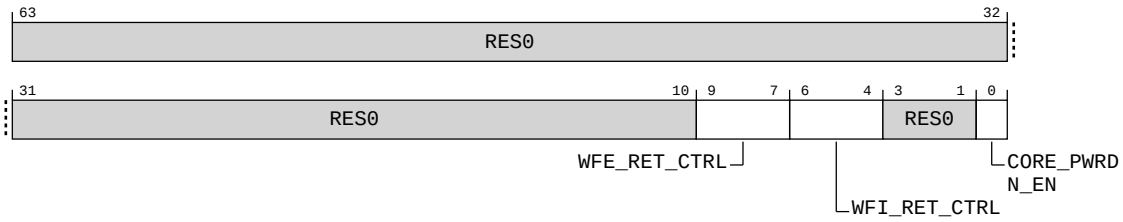


Table A-107: IMP_CPUPWRCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:10]	**RES0**	Reserved	**RES0**
[9:7]	WFE_RET_CTRL	Wait for Event retention control. The possible values are: **0b000** Dynamic retention is disabled. **0b001** 2 system counter ticks are required before retention entry. **0b010** 8 system counter ticks are required before retention entry. **0b011** 32 system counter ticks are required before retention entry. **0b100** 64 system counter ticks are required before retention entry. **0b101** 128 system counter ticks are required before retention entry. **0b110** 256 system counter ticks are required before retention entry. **0b111** 512 system counter ticks are required before retention entry.	0b000

Bits	Name	Description	Reset
[6:4]	WFI_RET_CTRL	Wait for Interrupt retention control. The possible values are: **0b000** Dynamic retention is disabled. **0b001** 2 system counter ticks are required before retention entry. **0b010** 8 system counter ticks are required before retention entry. **0b011** 32 system counter ticks are required before retention entry. **0b100** 64 system counter ticks are required before retention entry. **0b101** 128 system counter ticks are required before retention entry. **0b110** 256 system counter ticks are required before retention entry. **0b111** 512 system counter ticks are required before retention entry.	0b000
[3:1]	**RES0**	Reserved	**RES0**
[0]	CORE_PWRDN_EN	Indicates to the power controller if the CPU wants to power down when it enters WFE/WFI state. The possible values are: **0b0** CPU does not want to power down when it enters WFE/WFI state. **0b1** CPU wants to power down when it enters WFE/WFI state.	0b0

Access

MRS <Xt>, S3_0_C15_C2_7

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b111

MSR S3_0_C15_C2_7, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b111

Accessibility

MRS <Xt>, S3_0_C15_C2_7

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

```
        X[t, 64] = IMP_CPUPWRCTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPWRCTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPWRCTLR_EL1;
```

```
MSR S3_0_C15_C2_7, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.PWREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.PWREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPWRCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.PWREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPWRCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUPWRCTLR_EL1 = X[t, 64];
```

A.4.15 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 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 A-41: AARCH64_IMP_ATCR_EL1 bit assignments

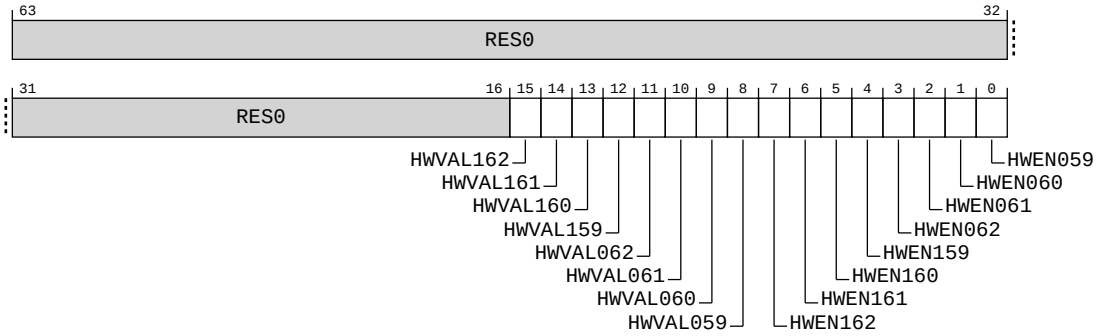


Table A-110: IMP_ATCR_EL1 bit descriptions

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

Access

`MRS <Xt>, S3_0_C15_C7_0`

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0111	0b000

`MSR S3_0_C15_C7_0, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0111	0b000

Accessibility

`MRS <Xt>, S3_0_C15_C7_0`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_ATCR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_ATCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_ATCR_EL1;
```

`MSR S3_0_C15_C7_0, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_ATCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    IMP_ATCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL1 = X[t, 64];
```

A.4.16 IMP_CPUACTLR5_EL1, CPU Auxiliary Control Register 5 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-42: AARCH64_IMP_CPUACTLR5_EL1 bit assignments

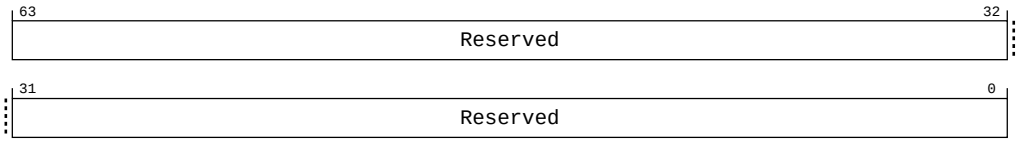


Table A-113: IMP_CPUACTLR5_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_0

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b000

MSR S3_0_C15_C8_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b000

Accessibility

MRS <Xt>, S3_0_C15_C8_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR5_EL1;
elsif PSTATE.EL == EL2 then
```

```
    X[t, 64] = IMP_CPUACTLR5_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR5_EL1;
```

```
MSR S3_0_C15_C8_0, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR5_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR5_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR5_EL1 = X[t, 64];
```

A.4.17 IMP_CPUACTLR6_EL1, CPU Auxiliary Control Register 6 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-43: AARCH64_IMP_CPUACTLR6_EL1 bit assignments

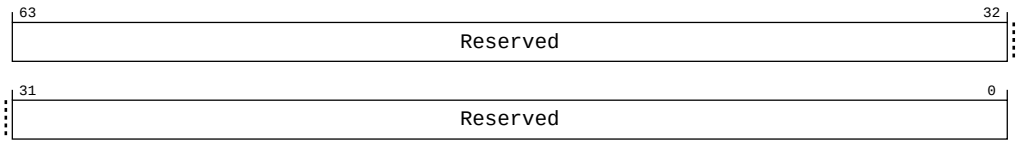


Table A-116: IMP_CPUACTLR6_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b001

MSR S3_0_C15_C8_1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b001

Accessibility

MRS <Xt>, S3_0_C15_C8_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR6_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR6_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR6_EL1;
```

MSR S3_0_C15_C8_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR6_EL1 = X[t, 64];
```

```
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR6_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR6_EL1 = X[t, 64];
```

A.4.18 IMP_CPUACTLR7_EL1, CPU Auxiliary Control Register 7 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-44: AARCH64_IMP_CPUACTLR7_EL1 bit assignments

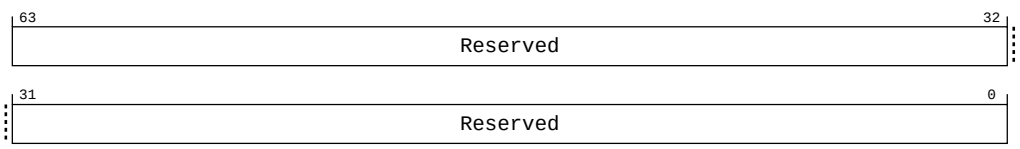


Table A-119: IMP_CPUACTLR7_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_2

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b010

MSR S3_0_C15_C8_2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b010

Accessibility

MRS <Xt>, S3_0_C15_C8_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR7_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR7_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR7_EL1;
```

MSR S3_0_C15_C8_2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR7_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR7_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR7_EL1 = X[t, 64];
```

A.4.19 IMP_CPUACTLR10_EL1, CPU Auxiliary Control Register 10 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-45: AARCH64_IMP_CPUACTLR10_EL1 bit assignments

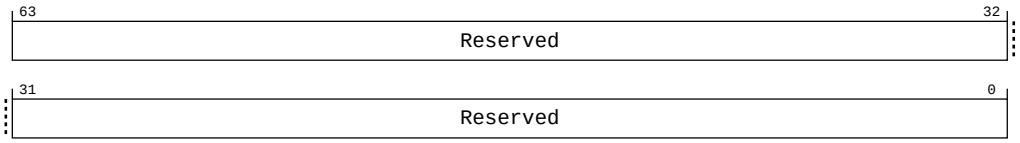


Table A-122: IMP_CPUACTLR10_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_3

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b011

MSR S3_0_C15_C8_3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b011

Accessibility

`MRS <Xt>, S3_0_C15_C8_3`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR10_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR10_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR10_EL1;
```

`MSR S3_0_C15_C8_3, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR10_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR10_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR10_EL1 = X[t, 64];
```

A.4.20 IMP_CPUACTLR8_EL1, CPU Auxiliary Control Register 8 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx 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-46: AARCH64_IMP_CPUACTLR8_EL1 bit assignments

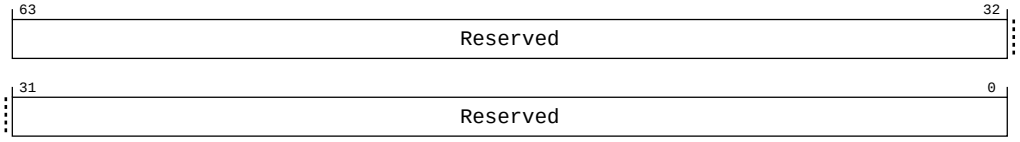


Table A-125: IMP_CPUACTLR8_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_5

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b101

MSR S3_0_C15_C8_5, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b101

Accessibility

MRS <Xt>, S3_0_C15_C8_5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR8_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR8_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR8_EL1;
```

```
MSR S3_0_C15_C8_5, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR8_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR8_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR8_EL1 = X[t, 64];
```

A.4.21 IMP_CPUACTLR9_EL1, CPU Auxiliary Control Register 9 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-47: AARCH64_IMP_CPUACTLR9_EL1 bit assignments

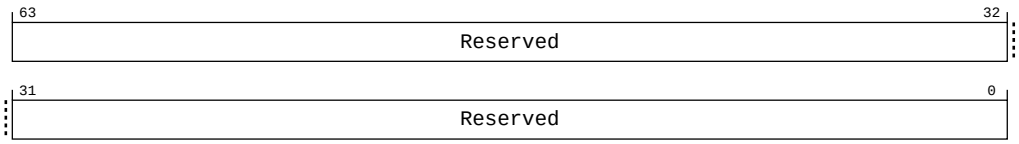


Table A-128: IMP_CPUACTLR9_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_6

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b110

MSR S3_0_C15_C8_6, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b110

Accessibility

MRS <Xt>, S3_0_C15_C8_6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR9_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR9_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR9_EL1;
```

MSR S3_0_C15_C8_6, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR9_EL1 = X[t, 64];
```

```
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR9_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR9_EL1 = X[t, 64];
```

A.4.22 IMP_CPUACTLR11_EL1, CPU Auxiliary Control Register 11 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-48: AARCH64_IMP_CPUACTLR11_EL1 bit assignments

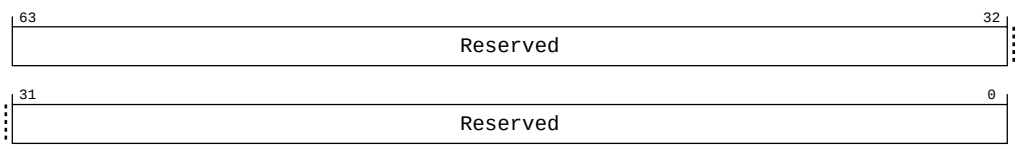


Table A-131: IMP_CPUACTLR11_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_7

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b111

MSR S3_0_C15_C8_7, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b111

Accessibility

MRS <Xt>, S3_0_C15_C8_7

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR11_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR11_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR11_EL1;
```

MSR S3_0_C15_C8_7, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR11_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR11_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR11_EL1 = X[t, 64];
```

A.4.23 IMP_CPUACTLR12_EL1, CPU Auxiliary Control Register 12 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-49: AARCH64_IMP_CPUACTLR12_EL1 bit assignments

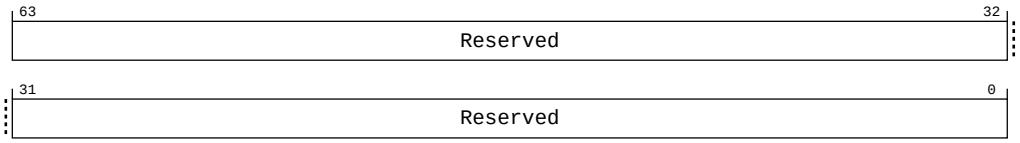


Table A-134: IMP_CPUACTLR12_EL1 bit descriptions

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

Access

```
MRS <Xt>, S3_0_C15_C13_0
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1101	0b000

```
MSR S3_0_C15_C13_0, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1101	0b000

Accessibility

`MRS <Xt>, S3_0_C15_C13_0`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR12_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR12_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR12_EL1;
```

`MSR S3_0_C15_C13_0, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR12_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR12_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR12_EL1 = X[t, 64];
```

A.4.24 IMP_CPUACTLR13_EL1, CPU Auxiliary Control Register 13 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx 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-50: AARCH64_IMP_CPUACTLR13_EL1 bit assignments

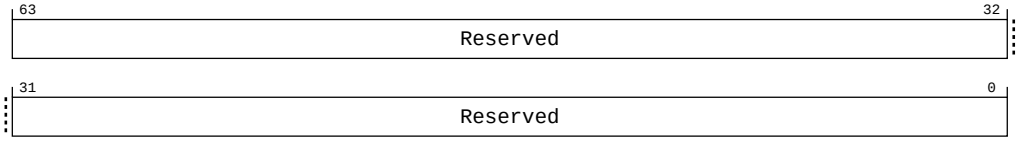


Table A-137: IMP_CPUACTLR13_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C13_1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1101	0b001

MSR S3_0_C15_C13_1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1101	0b001

Accessibility

MRS <Xt>, S3_0_C15_C13_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR13_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR13_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR13_EL1;
```

```
MSR S3_0_C15_C13_1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR13_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR13_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR13_EL1 = X[t, 64];
```

A.4.25 AIDR_EL1, Auxiliary ID Register

Provides **IMPLEMENTATION DEFINED** identification information.

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

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
```

Bit descriptions

Figure A-51: AARCH64_AIDR_EL1 bit assignments

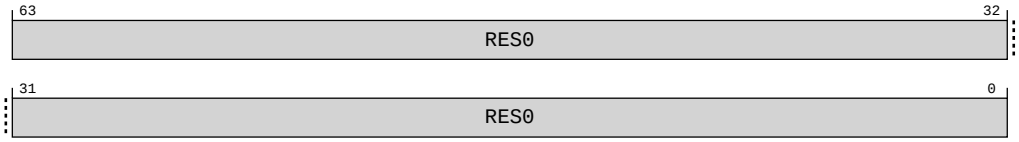


Table A-140: AIDR_EL1 bit descriptions

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

Access

`MRS <Xt>, AIDR_EL1`

op0	op1	CRn	CRm	op2
0b11	0b001	0b0000	0b0000	0b111

Accessibility

`MRS <Xt>, AIDR_EL1`

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = AIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AIDR_EL1;
```

A.4.26 FPCR, Floating-point Control Register

Controls floating-point behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-52: AARCH64_FPCR bit assignments

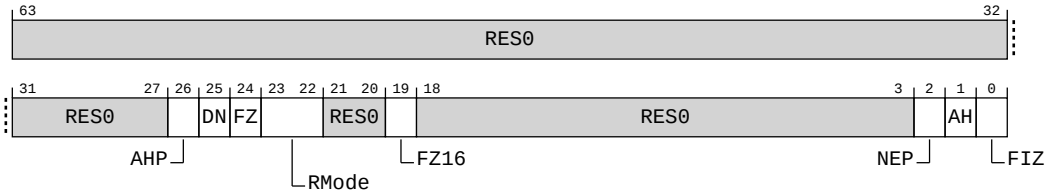


Table A-142: FPCR bit descriptions

Bits	Name	Description	Reset
[63:27]	RES0	Reserved	RES0
[26]	AHP	Alternative half-precision control bit. **0b0** IEEE half-precision format selected. **0b1** Alternative half-precision format selected.	x
[25]	DN	Default NaN use for NaN propagation. **0b0** NaN operands propagate through to the output of a floating-point operation. **0b1** Any operation involving one or more NaNs returns the Default NaN. This bit has no effect on the output of the FABS and FNEG instructions. This bit has no effect on the output of the FMAX, FMAXP, FMAXV, FMIN, FMINP, and FMINV instructions when FPCR.AH is 1.	x

Bits	Name	Description	Reset
[24]	FZ	Flushing denormalized numbers to zero control bit. **0b0** If FPCR.AH is 0, the flushing to zero of single-precision and double-precision denormalized inputs to, and outputs of, floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero. If FPCR.AH is 1, the flushing to zero of single-precision and double-precision denormalized outputs of floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero. **0b1** If FPCR.AH is 0, denormalized single-precision and double-precision inputs to, and outputs from, floating-point instructions are flushed to zero. If FPCR.AH is 1, denormalized single-precision and double-precision outputs from floating-point instructions are flushed to zero.	x
[23:22]	RMode	Rounding Mode control field. **0b00** Round to Nearest (RN) mode. **0b01** Round towards Plus Infinity (RP) mode. **0b10** Round towards Minus Infinity (RM) mode. **0b11** Round towards Zero (RZ) mode.	xx
[21:20]	**RES0**	Reserved	**RES0**
[19]	FZ16	Flushing denormalized numbers to zero control bit on half-precision data-processing instructions. **0b0** For some instructions, this bit disables flushing to zero of inputs and outputs that are half-precision denormalized numbers. **0b1** Flushing denormalized numbers to zero enabled. For some instructions that do not convert a half-precision input to a higher precision output, this bit enables flushing to zero of inputs and outputs that are half-precision denormalized numbers.	x
[18:3]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[2]	NEP	Controls how the output elements other than the lowest element of the vector are determined for Advanced SIMD scalar instructions. **0b0** Does not affect how the output elements other than the lowest are determined for Advanced SIMD scalar instructions. **0b1** The output elements other than the lowest are taken from the following registers: • For 3-input scalar versions of the FMLA (by element) and FMLS (by element) instructions, the <Hd>, <Sd>, or <Dd> register. • For 3-input versions of the FMADD, FMSUB, FNMADD, and FNMSUB instructions, the <Ha>, <Sa>, or <Da> register. • For 2-input scalar versions of the FACGE, FACGT, FCMEQ (register), FCMGE (register), and FCMGT (register) instructions, the <Hm>, <Sm>, or <Dm> register. • For 2-input scalar versions of the FABD, FADD (scalar), FDIV (scalar), FMAX (scalar), FMAXNM (scalar), FMIN (scalar), FMINNM (scalar), FMUL (by element), FMUL (scalar), FMULX (by element), FMULX, FNMUL (scalar), FRECPS, FRSQRTS, and FSUB (scalar) instructions, the <Hn>, <Sn>, or <Dn> register. • For 1-input scalar versions of the following instructions, the <Hd>, <Sd>, or <Dd> register: ◦ The (vector) versions of the FCVTAS, FCVTAU, FCVTMS, FCVTMU, FCVTNS, FCVTNU, FCVTPS, and FCVTPU instructions. ◦ The (vector, fixed-point) and (vector, integer) versions of the FCVTZS, FCVTZU, SCVTF, and UCVTF instructions. ◦ The (scalar) versions of the FABS, FNEG, FRINT32X, FRINT32Z, FRINT64X, FRINT64Z, FRINTA, FRINTI, FRINTM, FRINTN, FRINTP, FRINTX, FRINTZ, and FSQRT instructions. ◦ The (scalar, fixed-point) and (scalar, integer) versions of the SCVTF and UCVTF instructions. ◦ The BFCVT, FCVT, FCVTXN, FRECPE, FRECPX, and FRSQRTE instructions.	x
[1]	AH	Alternate Handling. Controls alternate handling of floating-point numbers. **0b1**	x
[0]	FIZ	Flush Inputs to Zero. Controls whether single-precision, double-precision and BFloat16 input operands that are denormalized numbers are flushed to zero. **0b0** The flushing to zero of single-precision and double-precision denormalized inputs to floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero. **0b1** Denormalized single-precision and double-precision inputs to most floating-point instructions flushed to zero.	x

Access

```
MRS <Xt>, FPCR
```

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b000

```
MSR FPCR, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b000

Accessibility

MRS <Xt>, FPCR

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !ELIsInHost(EL0) && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif ELIsInHost(EL0) && CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif ELIsInHost(EL2) && CPTR_EL2.FPEN IN {'x0'} then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && !ELIsInHost(EL2) && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif CPACR_EL1.FPEN IN {'x0'} then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && !ELIsInHost(EL2) && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif ELIsInHost(EL2) && CPTR_EL2.FPEN IN {'x0'} then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !ELIsInHost(EL2) && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif ELIsInHost(EL2) && CPTR_EL2.FPEN IN {'x0'} then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
```

MSR FPCR, <Xt>

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !ELIsInHost(EL0) && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif ELIsInHost(EL0) && CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif ELIsInHost(EL2) && CPTR_EL2.FPEN IN {'x0'} then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && !ELIsInHost(EL2) && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif CPACR_EL1.FPEN IN {'x0'} then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && !ELIsInHost(EL2) && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif ELIsInHost(EL2) && CPTR_EL2.FPEN IN {'x0'} then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !ELIsInHost(EL2) && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif ELIsInHost(EL2) && CPTR_EL2.FPEN IN {'x0'} then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
```

A.4.27 ACTLR_EL2, Auxiliary Control Register (EL2)

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

Note

Arm recommends the contents of this register are updated to apply to EL0 when the Effective value of HCR_EL2.{E2H, TGE} is {1, 1}, gaining configuration and control fields from the ACTLR_EL1. This avoids the need for software to manage the contents of these register when switching between a Guest OS and a Host OS.

Configurations

If EL2 is not implemented, this register is **RES0** from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 0000 0xxx 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 A-53: AARCH64_ACTLR_EL2 bit assignments

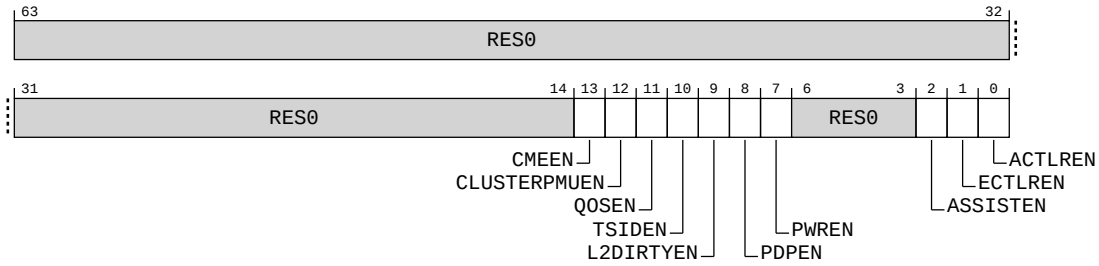


Table A-145: ACTLR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:14]	**RES0**	Reserved	**RES0**
[13]	CMEEN	C1-Scalable Matrix Extension 2 unit write enable. The possible values are: **0b0** IMP_CME* registers are not write-accessible from a lower Exception level. This is the reset value. **0b1** IMP_CME* registers are write-accessible from a lower Exception level if they are write-accessible from EL2	0b0
[12]	CLUSTERPMUEN	Performance Management Registers write enable. The possible values are: **0b0** CLUSTERPM* registers are not write-accessible from EL1. This is the reset value. **0b1** CLUSTERPM* registers are write-accessible from EL1 if they are write-accessible from EL2.	0b0
[11]	QOSEN	CPU Bus QoS Register write enable. The possible values are: **0b0** Register IMP_CPUBUSQOS_EL1 is not write-accessible from EL1. This is the reset value. **0b1** Register IMP_CPUBOSQOS_EL1 is write-accessible from EL1 if it is also write-accessible from EL2	0b0
[10]	TSIDEN	Thread Scheme ID Register write enable. The possible values are: **0b0** Register IMP_CLUSTERTHREADSID is not write-accessible from EL1. This is the reset value. **0b1** Register IMP_CLUSTERTHREADSID is write-accessible from EL1 if they are write-accessible from EL2	0b0
[9]	L2DIRTYEN	L2 Dirty Line Count Register write enable. The possible values are: **0b0** Register IMP_CPUL2DIRTYLNCT_EL1 is not read-accessible in EL1. This is the reset value. **0b1** Register IMP_CPUL2DIRTYLNCT_EL1 is read-accessible in EL1.	0b0
[8]	PDPEN	PDP control register write enable. Possible values of this bit are: **0b0** IMP_CPUPPMPDPCR_EL1 is not write-accessible in EL1. This is the reset value. **0b1** IMP_CPUPPMPDPCR_EL1 is write-accessible in EL1 if it is write-accessible from EL2.	0b0

Bits	Name	Description	Reset
[7]	PWREN	Power Control Registers write enable. The possible values are: **0b0** Registers IMP_CPUPWRCTLR, IMP_CLUSTERPWRCTLR, IMP_CLUSTERPWRDN, IMP_CLUSTERPWRSTAT, IMP_CLUSTERL3HIT and IMP_CLUSTERL3MISS are not write accessible from EL1. This is the reset value. **0b1** Registers IMP_CPUPWRCTLR, IMP_CLUSTERPWRCTLR, IMP_CLUSTERPWRDN, IMP_CLUSTERPWRSTAT, IMP_CLUSTERL3HIT and IMP_CLUSTERL3MISS are write-accessible from EL1 if they are write-accessible from EL2	0b0
[6:3]	**RES0**	Reserved	**RES0**
[2]	ASSISTEN	Assist feature enable. Possible values of this bit are: **0b0** IMP_CPUSYNCASSISTCR_EL1 is not write-accessible from EL1. This is the reset value. **0b1** IMP_CPUSYNCASSISTCR_EL1 is write-accessible from EL1 if it is write-accessible from EL2.	0b0
[1]	ECTLREN	Extended Control Registers write enable. The possible values are: **0b0** IMP_CPUECTLR*_EL1 and CLUSTERECTLR_EL1 are not write-accessible from EL1. This is the reset value. **0b1** IMP_CPUECTLR*_EL1 and CLUSTERECTLR_EL1 are write-accessible from EL1 if they are write-accessible from EL2.	0b0
[0]	ACTLREN	Auxiliary Control Registers write enable. The possible values are: **0b0** IMP_CPUACTLR*_EL1 and CLUSTERACTLR are not write-accessible from EL1. This is the reset value. **0b1** IMP_CPUACTLR*_EL1 and CLUSTERACTLR are write-accessible from EL1 if they are write-accessible from EL2	0b0

Access

`MRS <Xt>, ACTLR_EL2`

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0000	0b001

`MSR ACTLR_EL2, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0000	0b001

`MRS <Xt>, ACTLR_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b001

MSR ACTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b001

Accessibility

MRS <Xt>, ACTLR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ACTLR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ACTLR_EL2;
```

MSR ACTLR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    ACTLR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ACTLR_EL2 = X[t, 64];
```

MRS <Xt>, ACTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'1x1'} then
        if EffectiveHCR_EL2_NVx() == '101' && boolean IMPLEMENTATION_DEFINED
 "IMPLEMENTED_ACTLR_ELx accessor behavior" then
            X[t, 64] = ACTLR_EL1;
        else
            X[t, 64] = NVMem[0x118];
    else
        X[t, 64] = ACTLR_EL1;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) && boolean IMPLEMENTATION_DEFINED "IMPLEMENTED_ACTLR_ELx
 accessor behavior" then
        X[t, 64] = ACTLR_EL2;
    else
        X[t, 64] = ACTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ACTLR_EL1;
```

```
MSR ACTLR_EL1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'1x1'} then
        if EffectiveHCR_EL2_NVx() == '101' && boolean IMPLEMENTATION_DEFINED
 "IMPLEMENTED_ACTLR_ELx accessor behavior" then
            ACTLR_EL1 = X[t, 64];
        else
            NVMem[0x118] = X[t, 64];
    else
        ACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) && boolean IMPLEMENTATION_DEFINED "IMPLEMENTED_ACTLR_ELx
 accessor behavior" then
        ACTLR_EL2 = X[t, 64];
    else
        ACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ACTLR_EL1 = X[t, 64];
```

A.4.28 HACR_EL2, Hypervisor Auxiliary Control Register

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


Note

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

Configurations

If EL2 is not implemented, this register is **RES0** from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic System Control

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-54: AARCH64_HACR_EL2 bit assignments

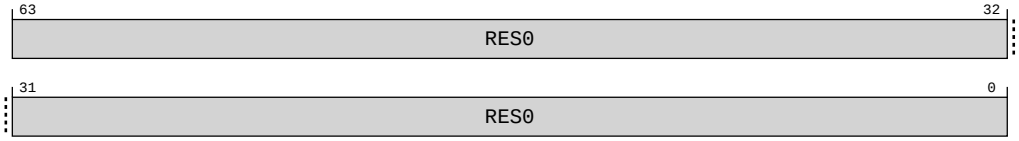


Table A-150: HACR_EL2 bit descriptions

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

Access

MRS <Xt>, HACR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b111

MSR HACR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b111

Accessibility

MRS <Xt>, HACR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = HACR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HACR_EL2;
```

MSR HACR_EL2, <Xt>

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

```
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HACR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HACR_EL2 = X[t, 64];
```

A.4.29 AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2)

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

Configurations

If EL2 is not implemented, this register is **RES0** from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic System Control

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-55: AARCH64_AFSR0_EL2 bit assignments

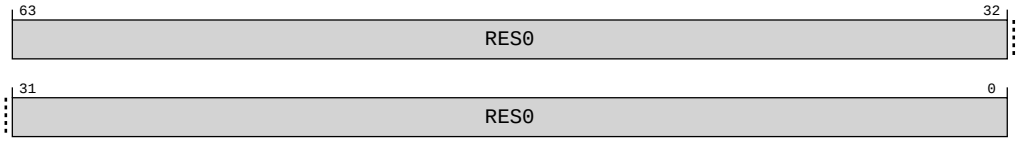


Table A-153: AFSR0_EL2 bit descriptions

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

Access

MRS <Xt>, AFSR0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b000

MSR AFSR0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b000

MRS <Xt>, AFSR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

MSR AFSR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

Accessibility

When the Effective value of HCR_EL2.E2H is 1, without explicit synchronization, accesses from EL2 using the accessor name `AFSR0_EL2` or `AFSR0_EL1` are not guaranteed to be ordered with respect to accesses using the other accessor name.

MRS <Xt>, AFSR0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AFSR0_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR0_EL2;
```

MSR AFSR0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
```

```
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AFSR0_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR0_EL2 = X[t, 64];
```

```
MRS <Xt>, AFSR0_EL1
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        X[t, 64] = NVMem[0x128];
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        X[t, 64] = AFSR0_EL2;
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR0_EL1;
```

```
MSR AFSR0_EL1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        NVMem[0x128] = X[t, 64];
    else
        AFSR0_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        AFSR0_EL2 = X[t, 64];
    else
        AFSR0_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t, 64];
```

A.4.30 AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2)

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

Configurations

If EL2 is not implemented, this register is **RES0** from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic System Control

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-56: AARCH64_AFSR1_EL2 bit assignments

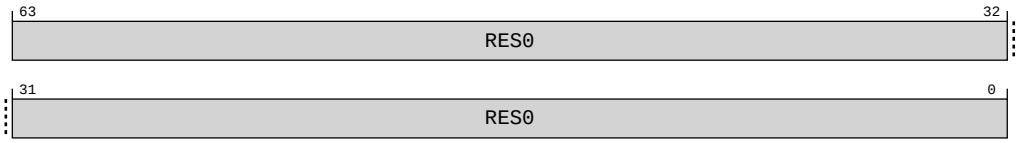


Table A-158: AFSR1_EL2 bit descriptions

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

Access

`MRS <Xt>, AFSR1_EL2`

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b001

`MSR AFSR1_EL2, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b001

`MRS <Xt>, AFSR1_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

MSR AFSR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

Accessibility

When the Effective value of HCR_EL2.E2H is 1, without explicit synchronization, accesses from EL2 using the accessor name AFSR1_EL2 or AFSR1_EL1 are not guaranteed to be ordered with respect to accesses using the other accessor name.

MRS <Xt>, AFSR1_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AFSR1_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR1_EL2;
```

MSR AFSR1_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AFSR1_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR1_EL2 = X[t, 64];
```

MRS <Xt>, AFSR1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        X[t, 64] = NVMem[0x130];
    else
        X[t, 64] = AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        X[t, 64] = AFSR1_EL2;
    else
        X[t, 64] = AFSR1_EL1;
```

```
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR1_EL1;
```

```
MSR AFSR1_EL1, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        NVMem[0x130] = X[t, 64];
    else
        AFSR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        AFSR1_EL2 = X[t, 64];
    else
        AFSR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR1_EL1 = X[t, 64];
```

A.4.31 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by MAIR_EL2.

Configurations

If EL2 is not implemented, this register is **RES0** from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic System Control

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-57: AARCH64_AMAIR_EL2 bit assignments

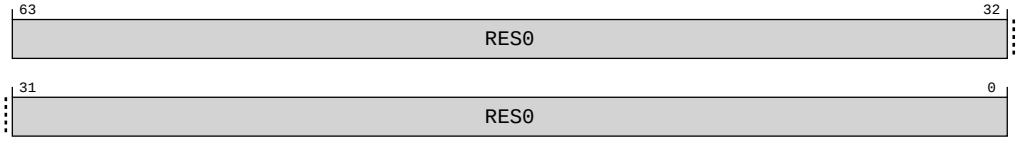


Table A-163: AMAIR_EL2 bit descriptions

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

Access

MRS <Xt>, AMAIR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0011	0b000

MSR AMAIR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0011	0b000

MRS <Xt>, AMAIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000

MSR AMAIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000

Accessibility

When the Effective value of HCR_EL2.E2H is 1, without explicit synchronization, accesses from EL2 using the accessor name AMAIR_EL2 or AMAIR_EL1 are not guaranteed to be ordered with respect to accesses using the other accessor name.

MRS <Xt>, AMAIR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AMAIR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR_EL2;
```

MSR AMAIR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AMAIR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AMAIR_EL2 = X[t, 64];
```

MRS <Xt>, AMAIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        X[t, 64] = NVMem[0x148];
    else
        X[t, 64] = AMAIR_EL1;
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        X[t, 64] = AMAIR_EL2;
    else
        X[t, 64] = AMAIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR_EL1;
```

MSR AMAIR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'111'} then
        NVMem[0x148] = X[t, 64];
    else
        AMAIR_EL1 = X[t, 64];
```

```
elsif PSTATE.EL == EL2 then
    if ELIsInHost(EL2) then
        AMAIR_EL2 = X[t, 64];
    else
        AMAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AMAIR_EL1 = X[t, 64];
```

A.4.32 IMP_ATCR_EL2, CPU Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 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 A-58: AARCH64_IMP_ATCR_EL2 bit assignments

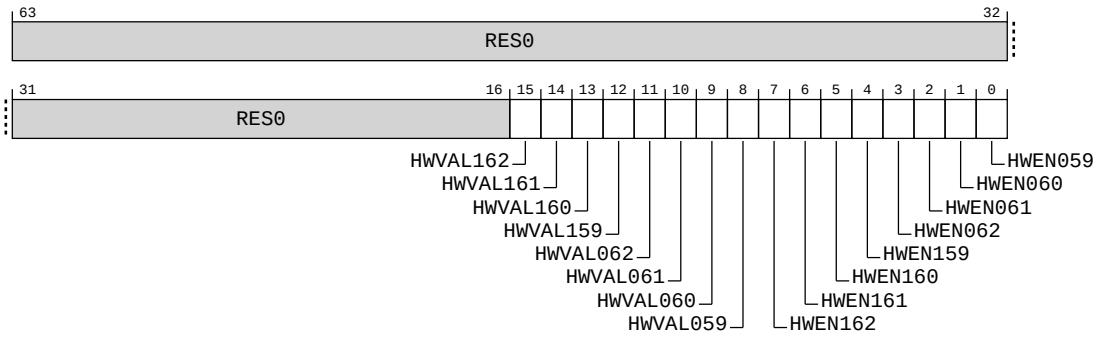


Table A-168: IMP_ATCR_EL2 bit descriptions

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

Access

```
MRS <Xt>, S3_4_C15_C7_0
```

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0111	0b000

```
MSR S3_4_C15_C7_0, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0111	0b000

Accessibility

```
MRS <Xt>, S3_4_C15_C7_0
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_ATCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_ATCR_EL2;
```

```
MSR S3_4_C15_C7_0, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_ATCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL2 = X[t, 64];
```

A.4.33 IMP_AVTCR_EL2, CPU Virtualization Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 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 A-59: AARCH64_IMP_AVTCR_EL2 bit assignments

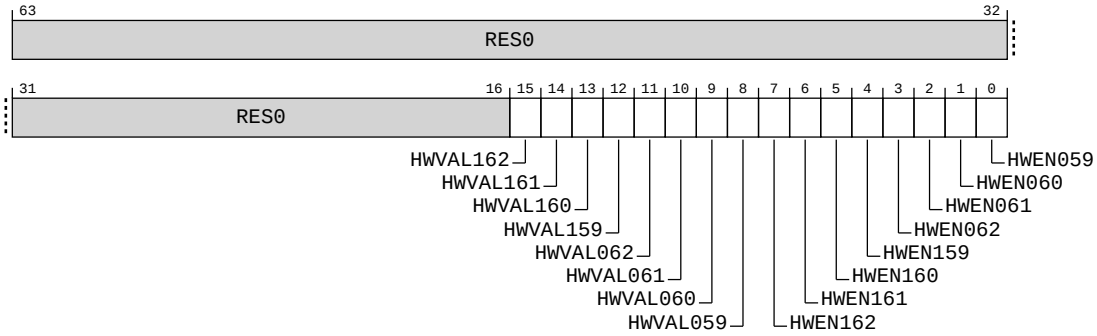


Table A-171: IMP_AVTCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN162 is set.	0b0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN161 is set.	0b0
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN160 is set.	0b0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN159 is set.	0b0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN062 is set.	0b0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN061 is set.	0b0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN060 is set.	0b0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN059 is set.	0b0
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0

Access

MRS <Xt>, S3_4_C15_C7_1

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0111	0b001

MSR S3_4_C15_C7_1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0111	0b001

Accessibility

MRS <Xt>, S3_4_C15_C7_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_AVTCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_AVTCR_EL2;
```

MSR S3_4_C15_C7_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_AVTCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_AVTCR_EL2 = X[t, 64];
```

A.4.34 ACTLR_EL3, Auxiliary Control Register (EL3)

Provides **IMPLEMENTATION DEFINED** configuration and control options for EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 0000 0xxx 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 A-60: AARCH64_ACTLR_EL3 bit assignments

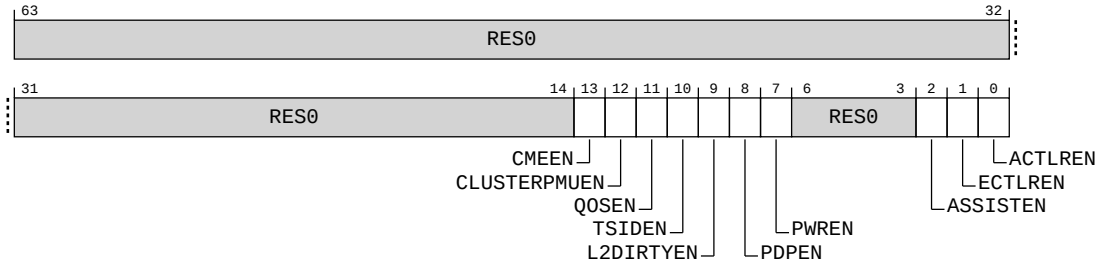


Table A-174: ACTLR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:14]	**RES0**	Reserved	**RES0**
[13]	CMEEN	C1-Scalable Matrix Extension 2 unit write enable. The possible values are: **0b0** AM* registers are not write-accessible from EL2 and EL2 and EL1. This is the reset value. **0b1** AM* registers are write-accessible from EL2 and EL2 and EL1	0b0
[12]	CLUSTERPMUEN	Performance Management Registers write enable. The possible values are: **0b0** IMP_CLUSTERPM* registers are not write-accessible from EL2 and EL1. This is the reset value. **0b1** IMP_CLUSTERPM* registers are write-accessible from EL2 and EL1 if they are write-accessible from EL2.	0b0

Bits	Name	Description	Reset
[11]	QOSEN	CPU Bus QoS Register write enable. The possible values are: **0b0** Register IMP_CPUBUSQOS_EL1 is not write-accessible EL2 and EL1. This is the reset value. **0b1** Register IMP_CPUBOSQOS_EL1 is write-accessible from EL2, and EL1 (if write-accessible from EL2)	0b0
[10]	TSIDEN	Thread Scheme ID Register write enable. The possible values are: **0b0** Register IMP_CLUSTERTHREADSID is not write-accessible from EL2 and EL1. This is the reset value. **0b1** Register IMP_CLUSTERTHREADSID is write-accessible from EL2 and EL1 if they are write-accessible from EL2	0b0
[9]	L2DIRTYEN	L2 Dirty Line Count Register write enable. The possible values are: **0b0** Register IMP_CPUL2DIRTYLNCT_EL1 is not read-accessible in EL2 and EL1. This is the reset value. **0b1** Register IMP_CPUL2DIRTYLNCT_EL1 is read-accessible in EL2 and EL1.	0b0
[8]	PDPEN	PDP control register write enable. Possible values of this bit are: **0b0** IMP_CPUPPMPDPCR_EL1 is not write-accessible from EL2 and EL1. This is the reset value. **0b1** IMP_CPUPPMPDPCR_EL1 is write-accessible from EL2 and EL1 if it is write-accessible from EL2	0b0
[7]	PWREN	Power Control Registers write enable. The possible values are: **0b0** Registers IMP_CPUPWRCTLR, IMP_CLUSTERPWRCTLR, IMP_CLUSTERPWRDN, IMP_CLUSTERPWRSTAT, IMP_CLUSTERL3HIT and IMP_CLUSTERL3MISS are not write accessible from EL2 and EL1. This is the reset value. **0b1** Registers IMP_CPUPWRCTLR, IMP_CLUSTERPWRCTLR, IMP_CLUSTERPWRDN, IMP_CLUSTERPWRSTAT, IMP_CLUSTERL3HIT and IMP_CLUSTERL3MISS are write-accessible from EL2 and EL1 if they are write-accessible from EL2	0b0
[6:3]	**RES0**	Reserved	**RES0**
[2]	ASSISTEN	Assist feature enable. Possible values of this bit are: **0b0** IMP_CPUSYNCASSISTCR_EL1 is not write-accessible from EL2 and EL1. This is the reset value. **0b1** IMP_CPUSYNCASSISTCR_EL1 is write-accessible from EL2 and EL1 if it is write-accessible from EL2	0b0

Bits	Name	Description	Reset
[1]	ECTLREN	Extended Control Registers write enable. The possible values are: **0b0** IMP_CPUECTLR*_EL2 and EL1 and IMP_CLUSTERECTLR_EL2 and EL1 are not write-accessible from EL2 and EL1. This is the reset value. **0b1** IMP_CPUECTLR*_EL2 and EL1 and IMP_CLUSTERECTLR_EL2 and EL1 are write-accessible from EL2 and EL1 if they are write-accessible from EL2.	0b0
[0]	ACTLREN	Auxiliary Control Registers write enable. The possible values are: **0b0** IMP_CPUACTLR*_EL2 and EL1 and IMP_CLUSTERACTLR are not write-accessible from EL2 and EL1. This is the reset value. **0b1** IMP_CPUACTLR*_EL2 and EL1 and IMP_CLUSTERACTLR are write-accessible from EL2 and EL1 if they are write-accessible from EL2	0b0

Access

MRS <Xt>, ACTLR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0000	0b001

MSR ACTLR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0000	0b001

Accessibility

MRS <Xt>, ACTLR_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ACTLR_EL3;
```

MSR ACTLR_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    ACTLR_EL3 = X[t, 64];
```

A.4.35 AFSR0_EL3, Auxiliary Fault Status Register 0 (EL3)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-61: AARCH64_AFSR0_EL3 bit assignments

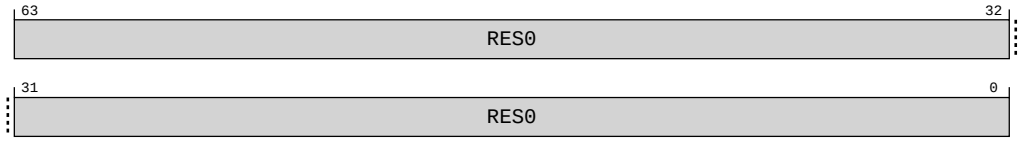


Table A-177: AFSR0_EL3 bit descriptions

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

Access

```
MRS <Xt>, AFSR0_EL3
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0001	0b000

```
MSR AFSR0_EL3, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0001	0b000

Accessibility

```
MRS <Xt>, AFSR0_EL3
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR0_EL3;
```

```
MSR AFSR0_EL3, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AFSR0_EL3 = X[t, 64];
```

A.4.36 AFSR1_EL3, Auxiliary Fault Status Register 1 (EL3)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-62: AARCH64_AFSR1_EL3 bit assignments

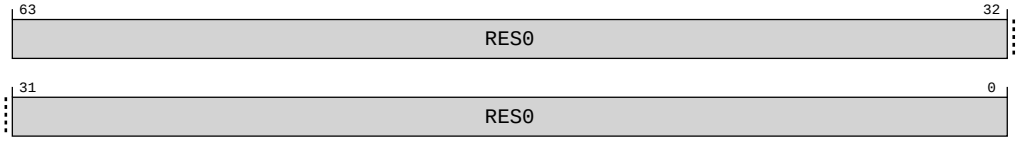


Table A-180: AFSR1_EL3 bit descriptions

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

Access

```
MRS <Xt>, AFSR1_EL3
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0001	0b001

```
MSR AFSR1_EL3, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0001	0b001

Accessibility

```
MRS <Xt>, AFSR1_EL3
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR1_EL3;
```

```
MSR AFSR1_EL3, <Xt>
```

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

```
elsif PSTATE.EL == EL3 then
    AFSR1_EL3 = X[t, 64];
```

A.4.37 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register (EL3)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by MAIR_EL3.

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-63: AARCH64_AMAIR_EL3 bit assignments

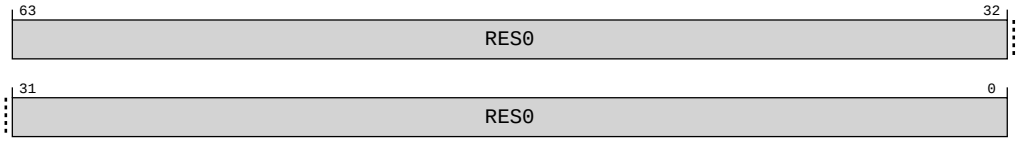


Table A-183: AMAIR_EL3 bit descriptions

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

Access

```
MRS <Xt>, AMAIR_EL3
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0011	0b000

```
MSR AMAIR_EL3, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0011	0b000

Accessibility

```
MRS <Xt>, AMAIR_EL3
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR_EL3;
```

```
MSR AMAIR_EL3, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AMAIR_EL3 = X[t, 64];
```

A.4.38 RMR_EL3, Reset Management Register (EL3)

If EL3 is implemented and this register is implemented:

- A write to the register at EL3 can request a Warm reset.
- If EL3 can use all Execution states, this register specifies the Execution state that the PE boots into on a Warm reset.

Configurations

When EL3 is implemented:

- If EL3 can use all Execution states then this register must be implemented.
- In a AArch64 only implementation it is **IMPLEMENTATION DEFINED** whether the register is implemented.

Otherwise, direct accesses to RMR_EL3 are **UNDEFINED**.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx01
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
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-64: AARCH64_RMR_EL3 bit assignments

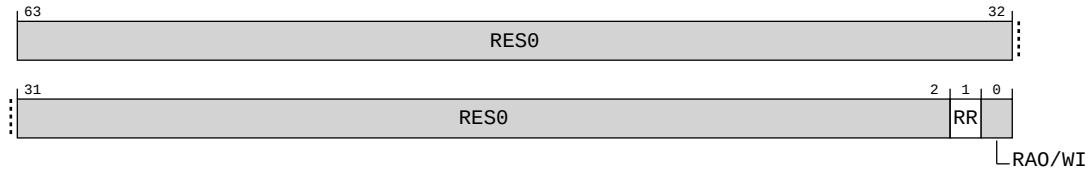


Table A-186: RMR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:2]	**RES0**	Reserved	**RES0**
[1]	RR	Reset Request. Setting this bit to 1 requests a Warm reset.	0b0
[0]	**RAO/WI**	Reserved	**RAO/WI**

Access

MRS <Xt>, RMR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b0000	0b010

MSR RMR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b0000	0b010

Accessibility

MRS <Xt>, RMR_EL3

```
if PSTATE.EL == EL3 then
    X[t, 64] = RMR_EL3;
else
    UNDEFINED;
```

MSR RMR_EL3, <Xt>

```
if PSTATE.EL == EL3 then
    RMR_EL3 = X[t, 64];
else
    UNDEFINED;
```

A.4.39 IMP_CPUL2SDIRTYLNCT_EL3, CPU L2 Secure Dirty Line Count Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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
```

Bit descriptions

Figure A-65: AARCH64_IMP_CPUL2SDIRTYLNCT_EL3 bit assignments

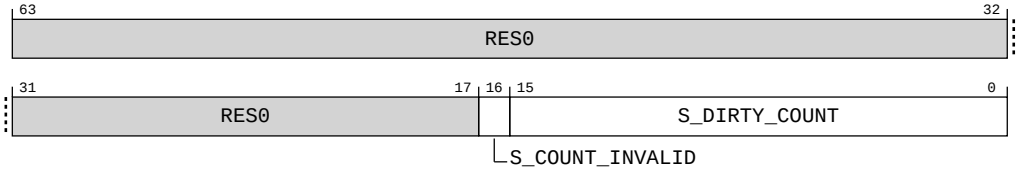


Table A-189: IMP_CPUL2SDIRTYLNCT_EL3 bit descriptions

Bits	Name	Description	Reset
[63:17]	RES0	Reserved	RES0
[16]	S_COUNT_INVALID	Indicates the secure dirty count is invalid. Reset value is 'b0	0b0
[15:0]	S_DIRTY_COUNT	Number of dirty secure lines in the L2. Reset value is 'h0000	0x0000

Access

`MRS <Xt>, S3_6_C15_C2_3`

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b011

Accessibility

`MRS <Xt>, S3_6_C15_C2_3`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUL2SDIRTYLNCT_EL3;
```

A.4.40 IMP_CPUACTLR_EL3, CPU Auxiliary Control Register (EL3)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-66: AARCH64_IMP_CPUACTLR_EL3 bit assignments

Table A-191: IMP_CPUACTLR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C4_0

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0100	0b000

MSR S3_6_C15_C4_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0100	0b000

Accessibility

MRS <Xt>, S3_6_C15_C4_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR_EL3;
```

MSR S3_6_C15_C4_0, <Xt>

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

```
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR_EL3 = X[t, 64];
```

A.4.41 IMP_ATCR_EL3, CPU Auxiliary Translation Control Register (EL3)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 xxxx 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 A-67: AARCH64_IMP_ATCR_EL3 bit assignments

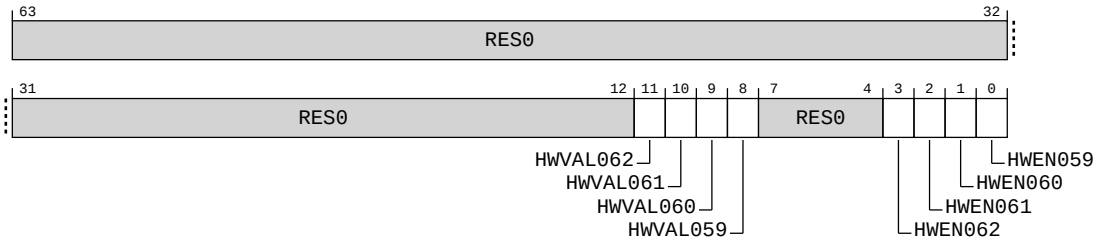


Table A-194: IMP_ATCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:12]	**RES0**	Reserved	**RES0**
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN062 is set.	0b0

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

Access

`MRS <Xt>, S3_6_C15_C7_0`

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0111	0b000

`MSR S3_6_C15_C7_0, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0111	0b000

Accessibility

`MRS <Xt>, S3_6_C15_C7_0`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_ATCR_EL3;
```

`MSR S3_6_C15_C7_0, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL3 = X[t, 64];
```

A.4.42 IMP_CPUPSELR_EL3, Selected Instruction Private Select Register

Selects the current instruction patch register for subsequent accesses to IMP_CPUPCR_EL3, IMP_CPUPOR_EL3, IMP_CPUPMR_EL3, IMP_CPUPOR2_EL3, IMP_CPUPMR2_EL3, and IMP_CPUPFR_EL3

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-68: AARCH64_IMP_CPUPSELR_EL3 bit assignments

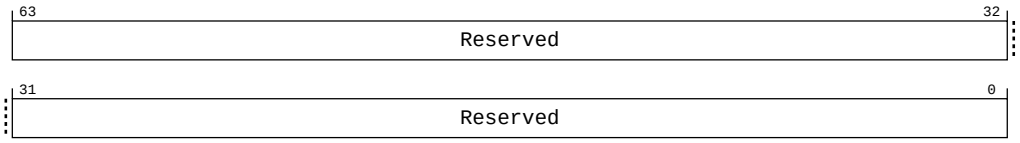


Table A-197: IMP_CPUPSELR_EL3 bit descriptions

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

Access

```
MRS <Xt>, S3_6_C15_C8_0
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b000

```
MSR S3_6_C15_C8_0, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b000

Accessibility

```
MRS <Xt>, S3_6_C15_C8_0
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPSELR_EL3;
```

```
MSR S3_6_C15_C8_0, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPSELR_EL3 = X[t, 64];
```

A.4.43 IMP_CPUPCR_EL3, Selected Instruction Private Control Register

Configures current Instruction Patch selected by IMP_CPUPSELR_EL3.SEL

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-69: AARCH64_IMP_CPUPCR_EL3 bit assignments

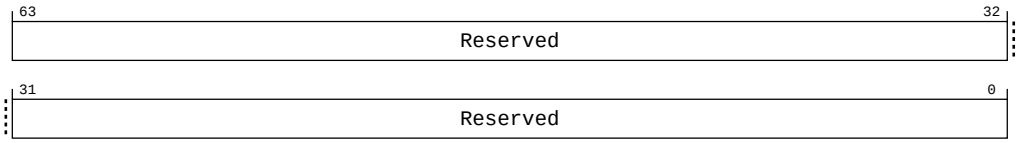


Table A-200: IMP_CPUPCR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_1

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b001

MSR S3_6_C15_C8_1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b001

Accessibility

MRS <Xt>, S3_6_C15_C8_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPCR_EL3;
```

MSR S3_6_C15_C8_1, <Xt>

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

```
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPCR_EL3 = X[t, 64];
```

A.4.44 IMP_CPUPOR_EL3, Selected Instruction Private Opcode Register

Opcode for current Instruction Patch selected by IMP_CPUPSELR_EL3.SEL

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-70: AARCH64_IMP_CPUPOR_EL3 bit assignments

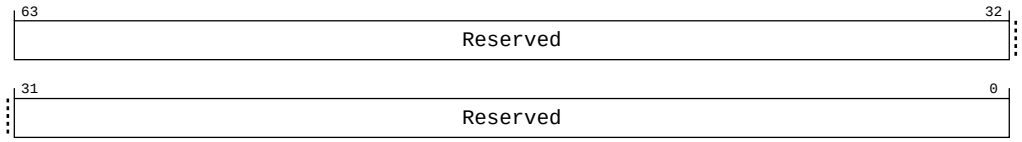


Table A-203: IMP_CPUPOR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_2

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b010

MSR S3_6_C15_C8_2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b010

Accessibility

MRS <Xt>, S3_6_C15_C8_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPOR_EL3;
```

MSR S3_6_C15_C8_2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPOR_EL3 = X[t, 64];
```

A.4.45 IMP_CPUPMR_EL3, Selected Instruction Private Mask Register

Mask for current Instruction Patch selected by IMP_CPUPSELR_EL3.SEL

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-71: AARCH64_IMP_CPUPMR_EL3 bit assignments

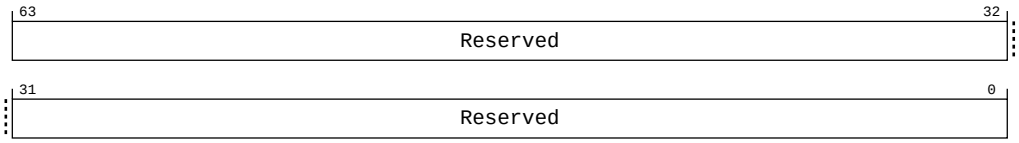


Table A-206: IMP_CPUPMR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b011

MSR S3_6_C15_C8_3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b011

Accessibility

MRS <Xt>, S3_6_C15_C8_3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPMR_EL3;
```

```
MSR S3_6_C15_C8_3, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPMR_EL3 = X[t, 64];
```

A.4.46 IMP_CPUPOR2_EL3, Selected Instruction Private Opcode Register 2

Opcode exclusion for current Instruction Patch selected by IMP_CPUPSELR_EL3.SEL

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-72: AARCH64_IMP_CPUPOR2_EL3 bit assignments

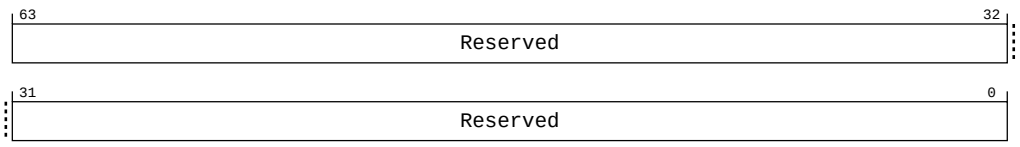


Table A-209: IMP_CPUPOR2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_4

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b100

MSR S3_6_C15_C8_4, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b100

Accessibility

MRS <Xt>, S3_6_C15_C8_4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPOR2_EL3;
```

MSR S3_6_C15_C8_4, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPOR2_EL3 = X[t, 64];
```

A.4.47 IMP_CPUPMR2_EL3, Selected Instruction Private Mask Register 2

Mask exclusion for current Instruction Patch selected by IMP_CPUPSELR_EL3.SEL

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-73: AARCH64_IMP_CPUPMR2_EL3 bit assignments

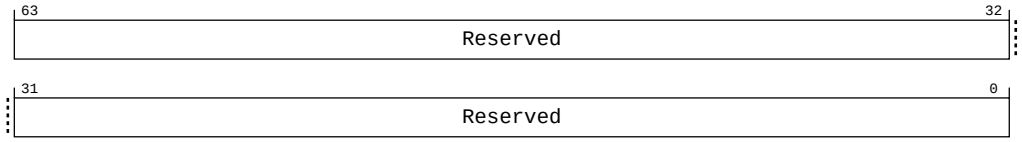


Table A-212: IMP_CPUPMR2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_5

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b101

MSR S3_6_C15_C8_5, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b101

Accessibility

MRS <Xt>, S3_6_C15_C8_5

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

```
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPMR2_EL3;
```

```
MSR S3_6_C15_C8_5, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPMR2_EL3 = X[t, 64];
```

A.4.48 IMP_CPUPFR_EL3, Selected Instruction Private Flag Register

Instruction Patch flags for current Instruction Patch selected by IMP_CPUPSELR_EL3.SEL

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-74: AARCH64_IMP_CPUPFR_EL3 bit assignments

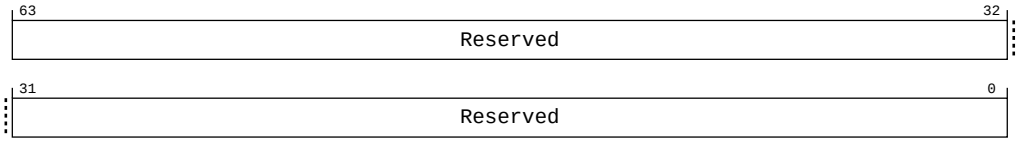


Table A-215: IMP_CPUPFR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_6

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b110

MSR S3_6_C15_C8_6, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b110

Accessibility

MRS <Xt>, S3_6_C15_C8_6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPFR_EL3;
```

MSR S3_6_C15_C8_6, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPFR_EL3 = X[t, 64];
```

A.5 AArch64 Identification Registers summary

The following summary table provides an overview of all Identification Registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-218: Identification Registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
MIDR_EL1	3	0	C0	C0	0	0x00000000411FD8C0	64-bit	Main ID Register
MPIDR_EL1	3	0	C0	C0	5	See individual bit resets.	64-bit	Multiprocessor Affinity Register
REVIDR_EL1	3	0	C0	C0	6	See individual bit resets.	64-bit	Revision ID Register
ID_PFR0_EL1	3	0	C0	C1	0	See individual bit resets.	64-bit	AArch32 Processor Feature Register 0
ID_PFR1_EL1	3	0	C0	C1	1	See individual bit resets.	64-bit	AArch32 Processor Feature Register 1
ID_DFR0_EL1	3	0	C0	C1	2	See individual bit resets.	64-bit	AArch32 Debug Feature Register 0
ID_AFR0_EL1	3	0	C0	C1	3	See individual bit resets.	64-bit	AArch32 Auxiliary Feature Register 0
ID_MMFR0_EL1	3	0	C0	C1	4	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 0
ID_MMFR1_EL1	3	0	C0	C1	5	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 1
ID_MMFR2_EL1	3	0	C0	C1	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 2
ID_MMFR3_EL1	3	0	C0	C1	7	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 3
ID_ISAR0_EL1	3	0	C0	C2	0	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 0
ID_ISAR1_EL1	3	0	C0	C2	1	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 1
ID_ISAR2_EL1	3	0	C0	C2	2	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 2
ID_ISAR3_EL1	3	0	C0	C2	3	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 3
ID_ISAR4_EL1	3	0	C0	C2	4	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 4
ID_ISAR5_EL1	3	0	C0	C2	5	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 5
ID_MMFR4_EL1	3	0	C0	C2	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 4
ID_ISAR6_EL1	3	0	C0	C2	7	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 6
MVFR0_EL1	3	0	C0	C3	0	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 0
MVFR1_EL1	3	0	C0	C3	1	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 1
MVFR2_EL1	3	0	C0	C3	2	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 2
ID_PFR2_EL1	3	0	C0	C3	4	See individual bit resets.	64-bit	AArch32 Processor Feature Register 2
ID_DFR1_EL1	3	0	C0	C3	5	See individual bit resets.	64-bit	Debug Feature Register 1
ID_MMFR5_EL1	3	0	C0	C3	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 5
ID_AA64PFR0_EL1	3	0	C0	C4	0	See individual bit resets.	64-bit	AArch64 Processor Feature Register 0
ID_AA64PFR1_EL1	3	0	C0	C4	1	See individual bit resets.	64-bit	AArch64 Processor Feature Register 1
ID_AA64PFR2_EL1	3	0	C0	C4	2	See individual bit resets.	64-bit	AArch64 Processor Feature Register 2

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ID_AA64ZFR0_EL1	3	0	C0	C4	4	See individual bit resets.	64-bit	SVE Feature ID Register 0
ID_AA64SMFR0_EL1	3	0	C0	C4	5	0x010050FF00000000	64-bit	SME Feature ID Register 0
ID_AA64DFR0_EL1	3	0	C0	C5	0	0x100F11F41030581A	64-bit	AArch64 Debug Feature Register 0
ID_AA64DFR1_EL1	3	0	C0	C5	1	0x0010000000000000	64-bit	AArch64 Debug Feature Register 1
ID_AA64AFR0_EL1	3	0	C0	C5	4	0x0000000000000000	64-bit	AArch64 Auxiliary Feature Register 0
ID_AA64AFR1_EL1	3	0	C0	C5	5	0x0000000000000000	64-bit	AArch64 Auxiliary Feature Register 1
ID_AA64ISAR0_EL1	3	0	C0	C6	0	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 0
ID_AA64ISAR1_EL1	3	0	C0	C6	1	0x0111121100311002	64-bit	AArch64 Instruction Set Attribute Register 1
ID_AA64ISAR2_EL1	3	0	C0	C6	2	0x0001000011115102	64-bit	AArch64 Instruction Set Attribute Register 2
ID_AA64MMFR0_EL1	3	0	C0	C7	0	0x2100022200101122	64-bit	AArch64 Memory Model Feature Register 0
ID_AA64MMFR1_EL1	3	0	C0	C7	1	0x1111112010312123	64-bit	AArch64 Memory Model Feature Register 1
ID_AA64MMFR2_EL1	3	0	C0	C7	2	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 2
ID_AA64MMFR3_EL1	3	0	C0	C7	3	0x1000000000000001	64-bit	AArch64 Memory Model Feature Register 3
MPAMIDR_EL1	3	0	C10	C4	4	0x240000010006003F	64-bit	MPAM ID Register (EL1)
IMP_CPUCFR_EL1	3	0	C15	C0	0	See individual bit resets.	64-bit	CPU Configuration Register
CCSIDR_EL1	3	1	C0	C0	0	See individual bit resets.	64-bit	Current Cache Size ID Register
CLIDR_EL1	3	1	C0	C0	1	0x00000054C3000123	64-bit	Cache Level ID Register
CCSIDR2_EL1	3	1	C0	C0	2	See individual bit resets.	64-bit	Current Cache Size ID Register 2
GMID_EL1	3	1	C0	C0	4	0x0000000000000004	64-bit	Multiple tag transfer ID Register
CSSELR_EL1	3	2	C0	C0	0	See individual bit resets.	64-bit	Cache Size Selection Register
CTR_EL0	3	3	C0	C0	1	0x000000049444C004	64-bit	Cache Type Register
DCZID_EL0	3	3	C0	C0	7	0x0000000000000004	64-bit	Data Cache Zero ID Register
VPIDR_EL2	3	4	C0	C0	0	See individual bit resets.	64-bit	Virtualization Processor ID Register
VMPIDR_EL2	3	4	C0	C0	5	See individual bit resets.	64-bit	Virtualization Multiprocessor ID Register

A.5.1 MIDR_EL1, Main ID Register

Provides identification information for the PE, including an implementer code for the device and a device ID number.

Configurations

AArch64 register MIDR_EL1 bits [31:0] are architecturally mapped to External register B.4.4 MIDR_EL1, Main ID Register on page 685 bits [31:0].

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0100 0001 0001 1111 1101 1000 1100 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-75: AARCH64_MIDR_EL1 bit assignments

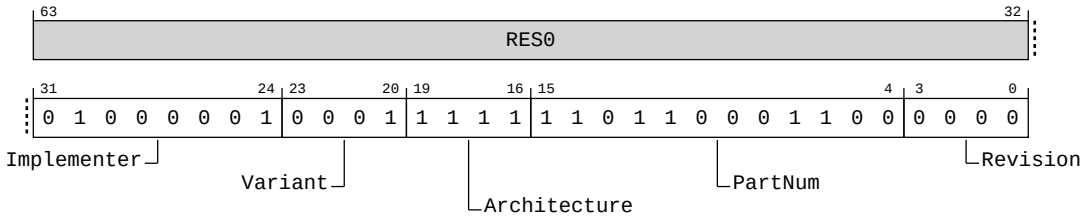


Table A-219: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	Implementer	Indicates the implementer code. This value is: **0x41** Arm Limited	0x41
[23:20]	Variant	Indicates the major revision of the product. **0b0001** r1p0	0b0001
[19:16]	Architecture	Architecture version. **0b1111** Architectural features are individually identified in the ID_* registers.	0b1111
[15:4]	PartNum	Primary Part Number for the device. On processors implemented by Arm, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently. **0xD8C** C1-Ultra	0xD8C
[3:0]	Revision	Indicates the minor revision of the product. **0b0000** r1p0	0b0000

Access

```
MRS <Xt>, MIDR_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b000

Accessibility

`MRS <Xt>, MIDR_EL1`

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.MIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() then
        X[t, 64] = VPIDR_EL2;
    else
        X[t, 64] = MIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MIDR_EL1;
```

A.5.2 MPIDR_EL1, Multiprocessor Affinity Register

In a multiprocessor system, provides an additional PE identification mechanism.

Configurations

In a uniprocessor system, Arm recommends that each Aff<n> field of this register returns a value of 0.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 xxxx xxxx 1000 0001 xxxx xxxx xxxx xxxx 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 A-76: AARCH64_MPIDR_EL1 bit assignments

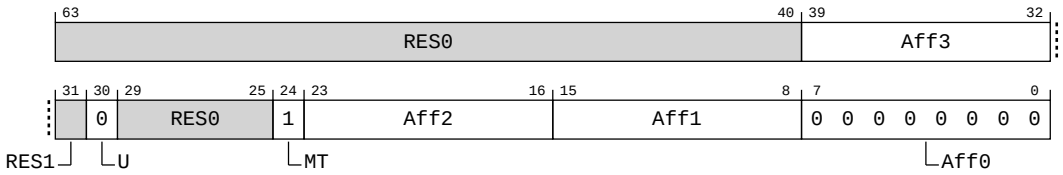


Table A-221: MPIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:40]	**RES0**	Reserved	**RES0**
[39:32]	Aff3	Affinity level 3. See the description of Aff0 for more information. Aff3 is not supported in AArch32 state.	`8{x}`
[31]	**RES1**	Reserved	**RES1**
[30]	U	Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system. **0b0** Processor is part of a multiprocessor system.	`0b0`
[29:25]	**RES0**	Reserved	**RES0**
[24]	MT	Indicates whether the lowest level of affinity consists of logical PEs that are implemented using an interdependent approach, such as multithreading. See the description of Aff0 for more information about affinity levels. **0b1** Performance of PEs with different affinity level 0 values, and the same values for affinity level 1 and higher, is very interdependent.	`0b1`
[23:16]	Aff2	Affinity level 2. See the description of Aff0 for more information.	`8{x}`
[15:8]	Aff1	Affinity level 1. See the description of Aff0 for more information. **0x00..0x0F**	The reset values can be the following: `0x00..0x0F`, respective to the value.
[7:0]	Aff0	Affinity level 0. The value of the MPIDR.{Aff2, Aff1, Aff0} or MPIDR_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole. **0x00**	`0x00`

Access

`MRS <Xt>, MPIDR_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b101

Accessibility

`MRS <Xt>, MPIDR_EL1`

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.MPIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() then
        X[t, 64] = VMPIDR_EL2;
    else
        X[t, 64] = MPIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MPIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPIDR_EL1;
```

A.5.3 REVIDR_EL1, Revision ID Register

The REVIDR_EL1 provides revision information, additional to MIDR_EL1, that identifies minor fixes (errata) which might be present in a specific implementation of the C1-Ultra core. Refer to the C1-Ultra Product Errata Notice (PEN) for information on how to interpret the values in this register.

Configurations

If REVIDR_EL1 has the same value as MIDR_EL1, then its contents have no significance.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 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-77: AARCH64_REVIDR_EL1 bit assignments

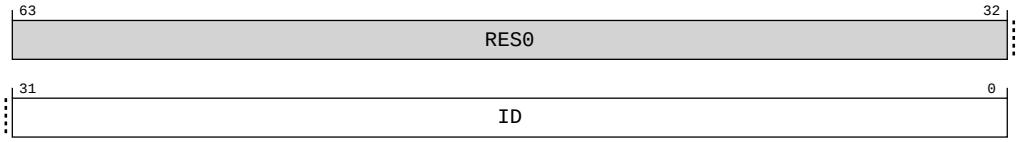


Table A-223: REVIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ID		32{x}

Access

MRS <Xt>, REVIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b110

Accessibility

MRS <Xt>, REVIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.REVIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = REVIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = REVIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = REVIDR_EL1;
```

A.5.4 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0

Provides additional information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

The external register EDPFR gives information from this register.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0001 0011 0000 0001 0001 0001 0001 0001 0010 xxxx 0001 0001 0001 0001 0001 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-78: AARCH64_ID_AA64PFR0_EL1 bit assignments

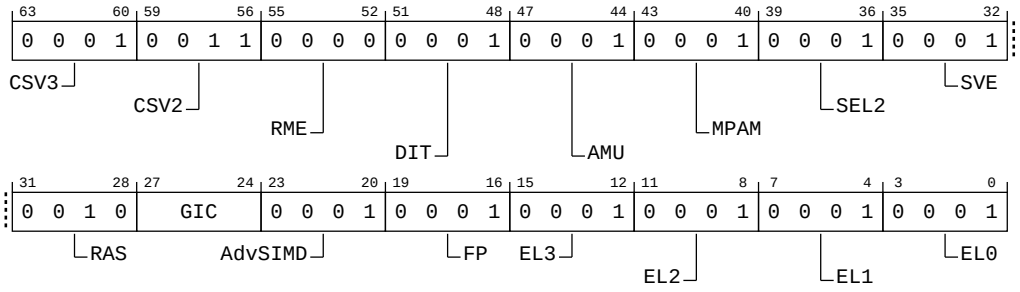


Table A-225: ID_AA64PFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	CSV3	Speculative use of faulting data. **0b0001** Data loaded under speculation with a permission or domain fault cannot be used to form an address, generate condition codes, or generate SVE predicate values to be used by other instructions in the speculative sequence. The execution timing of any other instructions in the speculative sequence is not a function of the data loaded under speculation.	0b0001
[59:56]	CSV2	Speculative use of out of context prediction resources. **0b0011** FEAT_CSV2_3 is implemented.	0b0011

Bits	Name	Description	Reset
[55:52]	RME	Realm Management Extension (RME). **0b0000** Realm Management Extension not implemented.	0b0000
[51:48]	DIT	Data Independent Timing. **0b0001** AArch64 provides the PSTATE.DIT mechanism to guarantee constant execution time of certain instructions.	0b0001
[47:44]	AMU	Indicates support for Activity Monitors Extension. **0b0001** FEAT_AMUv1 is implemented.	0b0001
[43:40]	MPAM	Indicates the major version number of support for the MPAM Extension. **0b0001** The major version number of the MPAM extension is 1.	0b0001
[39:36]	SEL2	Secure EL2. **0b0001** Secure EL2 is implemented.	0b0001
[35:32]	SVE	Scalable Vector Extension. **0b0001** SVE architectural state and programmers' model are implemented.	0b0001
[31:28]	RAS	RAS Extension version. **0b0010** As 0b0001, and adds support for: • If EL3 is implemented, FEAT_DoubleFault. • Additional ERXMISC<m>_EL1 System registers. • Additional System registers ERXPFGCDN_EL1, ERXPFGCTL_EL1, and ERXPFGF_EL1, and the SCR_EL3.FIEN and HCR_EL2.FIEN trap controls, to support the optional RAS Common Fault Injection Model Extension. Error records accessed through System registers conform to RAS System Architecture v1.1, which includes simplifications to ERR<n>STATUS and support for the optional RAS Timestamp and RAS Common Fault Injection Model Extensions.	0b0010
[27:24]	GIC	System register GIC CPU interface. **0b0000** GIC CPU interface system registers not implemented. This value applies when GICCDISABLE == TRUE. **0b0011** System register interface to version 4.1 of the GIC CPU interface is supported. This value applies when GICCDISABLE == FALSE.	The reset values can be the following: 0b0000, 0b0011, respective to the value.
[23:20]	AdvSIMD	Advanced SIMD. **0b0001** As for 0b0000, and also includes support for half-precision floating-point arithmetic.	0b0001

Bits	Name	Description	Reset
[19:16]	FP	Floating-point. **0b0001** As for 0b0000, and also includes support for half-precision floating-point arithmetic.	0b0001
[15:12]	EL3	EL3 Exception level handling. **0b0001** EL3 can be executed in AArch64 state only.	0b0001
[11:8]	EL2	EL2 Exception level handling. **0b0001** EL2 can be executed in AArch64 state only.	0b0001
[7:4]	EL1	EL1 Exception level handling. **0b0001** EL1 can be executed in AArch64 state only.	0b0001
[3:0]	EL0	EL0 Exception level handling. **0b0001** EL0 can be executed in AArch64 state only.	0b0001

Access

```
MRS <Xt>, ID_AA64PFR0_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b000

Accessibility

```
MRS <Xt>, ID_AA64PFR0_EL1
```

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64PFR0_EL1;
```

A.5.5 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1

Provides additional information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0001 0000 0001 0010 0000 0001 0000 xxxx 0010 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-79: AARCH64_ID_AA64PFR1_EL1 bit assignments

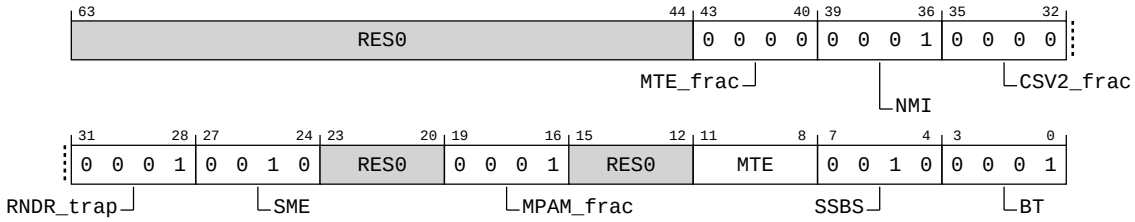


Table A-227: ID_AA64PFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:44]	**RES0**	Reserved	**RES0**
[43:40]	MTE_frac	Support for Asynchronous reporting of a Tag Check Fault. **0b0000** Asynchronous reporting of a Tag Check Fault is supported.	0b0000
[39:36]	NMI	Non-maskable Interrupt. Indicates support for Non-maskable interrupts. **0b0001** SCTLR_ELx.{SPINTMASK, NMI} and PSTATE.ALLINT with its associated instructions are supported.	0b0001

Bits	Name	Description	Reset
[35:32]	CSV2_frac	CSV2 fractional field. **0b0000** Either ID_AA64PFR0_EL1.CSV2 is not 0b0001, or the implementation does not disclose whether FEAT_CSV2_1p1 is implemented. FEAT_CSV2_1p2 is not implemented.	0b0000
[31:28]	RNDR_trap	Random Number trap to EL3 field. **0b0001** Trapping of RNDR and RNDRRS to EL3 is supported. SCR_EL3.TRNDR is present.	0b0001
[27:24]	SME	Scalable Matrix Extension. **0b0010** As 0b0001, plus the SME2 ZT0 register.	0b0010
[23:20]	**RES0**	Reserved	**RES0**
[19:16]	MPAM_frac	Indicates the minor version number of support for the MPAM Extension. **0b0001** The minor version number of the MPAM extension is 1.	0b0001
[15:12]	**RES0**	Reserved	**RES0**
[11:8]	MTE	Support for the Memory Tagging Extension. **0b0011** As 0b0010, except that support for FEAT_MTE_ASYNC is mandatory, and adds support for Asymmetric Tag Check Fault handling, identified as FEAT_MTE_ASYM_FAULT. This value applies when BROADCASTMTE == TRUE. **0b0001** Instruction-only Memory Tagging Extension is implemented. This value applies when BROADCASTMTE == FALSE.	The reset values can be the following: 0b0011, 0b0001, respective to the value.
[7:4]	SSBS	Speculative Store Bypassing controls in AArch64 state. **0b0010** As 0b0001, and adds the MSR and MRS instructions to directly read and write the PSTATE.SSBS field.	0b0010
[3:0]	BT	Branch Target Identification mechanism support in AArch64 state. **0b0001** The Branch Target Identification mechanism is implemented.	0b0001

Access

`MRS <Xt>, ID_AA64PFR1_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b001

Accessibility

`MRS <Xt>, ID_AA64PFR1_EL1`

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64PFR1_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64PFR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64PFR1_EL1;
```

A.5.6 ID_AA64PFR2_EL1, AArch64 Processor Feature Register 2

Provides additional information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, **RES0** from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 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-80: AARCH64_ID_AA64PFR2_EL1 bit assignments

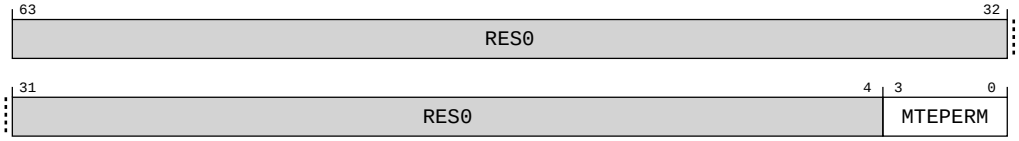


Table A-229: ID_AA64PFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3:0]	MTEPERM	Support for Allocation tag access permissions. **0b0000** Allocation tag access permissions are not supported. This value applies when BROADCASTMTE == FALSE. **0b0000** Allocation tag access permissions are not supported. This value applies when BROADCASTMTE == TRUE. **0b0001** Allocation tag access permissions are supported. **Note:** NoTagAccess is supported at stage 2 of translation only. This value applies when BROADCASTMTE == TRUE.	The reset values can be the following: 0b0000, 0b0000, 0b0001, respective to the value.

Access

MRS <Xt>, ID_AA64PFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b010

Accessibility

`MRS <Xt>, ID_AA64PFR2_EL1`

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64PFR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64PFR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64PFR2_EL1;
```

A.5.7 ID_AA64ZFR0_EL1, SVE Feature ID Register 0

Provides additional information about the implemented features of the AArch64 Scalable Vector Extension instruction set, when FEAT_SVE or FEAT_SME is implemented.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations


Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, **RES0** from EL1, EL2, and EL3.

If FEAT_SME is implemented and FEAT_SVE is not implemented, then SVE instructions can only be executed when the PE is in Streaming SVE mode and the instructions are legal for execution in Streaming SVE mode.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0000 0000 0000 0001 xxxx 0000 xxxx 0000 0000 0001 0001 0000 0000 xxxx 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
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-81: AARCH64_ID_AA64ZFR0_EL1 bit assignments

Table A-231: ID_AA64ZFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	**RES0**	Reserved	**RES0**
[59:56]	F64MM	Indicates support for the following SVE FP64 double-precision variant of the `FMMLA` instruction, the `LD1RO*` instructions, the 128-bit element variants of the SVE `TRN1`, `TRN2`, `UZP1`, `UZP2`, `ZIP1`, and `ZIP2` instructions. **0b0000** Double-precision matrix multiplication and related SVE instructions are not implemented.	0b0000
[55:52]	F32MM	Indicates support for the SVE FP32 single-precision floating-point matrix multiplication instruction. **0b0000** Single-precision matrix multiplication instruction is not implemented by this feature.	0b0000
[51:48]	**RES0**	Reserved	**RES0**
[47:44]	I8MM	Indicates support for the following SVE Int8 matrix multiplication instructions SVE `SMMLA`, `SUDOT`, `UMMLA`, `USMMLA`, and `USDOT`. **0b0001** The specified instructions are implemented.	0b0001
[43:40]	SM4	Indicates support for SVE SM4 instructions. **0b0001** SVE `SM4E` and `SM4EKEY` instructions are implemented. This value applies when CRYPTO == TRUE. **0b0000** SVE SM4 instructions are not implemented. This value applies when CRYPTO == FALSE.	The reset values can be the following: 0b0001, 0b0000, respective to the value.
[39:36]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[35:32]	SHA3	Indicates support for the SVE SHA3 instructions. **0b0001** SVE RAX1 instruction is implemented. This value applies when CRYPTO == TRUE. **0b0000** SVE SHA3 instructions are not implemented. This value applies when CRYPTO == FALSE.	The reset values can be the following: 0b0001, 0b0000, respective to the value.
[31:24]	**RES0**	Reserved	**RES0**
[23:20]	BF16	Indicates support for SVE BFloat16 instructions. **0b0001** SVE BFCVT, BFCVTNT, BFDOT, BFMLALB, BFMLALT, and BFMMLA instructions are implemented.	0b0001
[19:16]	BitPerm	Indicates support for the following SVE bit permute instructions SVE BDEP, BEXT, and BGRP. **0b0001** The specified instructions are implemented.	0b0001
[15:8]	**RES0**	Reserved	**RES0**
[7:4]	AES	Indicates support for SVE AES instructions. **0b0010** As 0b0001, and adds 64-bit source element variants of SVE PMULLB and PMULLT instructions. This value applies when CRYPTO == TRUE. **0b0000** SVE AES instructions are not implemented. This value applies when CRYPTO == FALSE.	The reset values can be the following: 0b0010, 0b0000, respective to the value.
[3:0]	SVEver	Indicates support for SVE instructions when FEAT_SME or FEAT_SVE is implemented. **0b0001** As 0b0000, and adds the mandatory SVE2 instructions.	0b0001

Access

MRS <Xt>, ID_AA64ZFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b100

Accessibility

MRS <Xt>, ID_AA64ZFR0_EL1

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

```
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ZFR0_EL1;
```

A.5.8 ID_AA64SMFR0_EL1, SME Feature ID Register 0

Provides information about the implemented features of the AArch64 Scalable Matrix Extension.

The fields in this register do not follow the standard ID scheme. See *Alternative ID scheme used for ID_AA64SMFR0_EL1* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, **RES0** from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0001 0000 0000 0101 0000 1111 1111 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 A-82: AARCH64_ID_AA64SMFR0_EL1 bit assignments

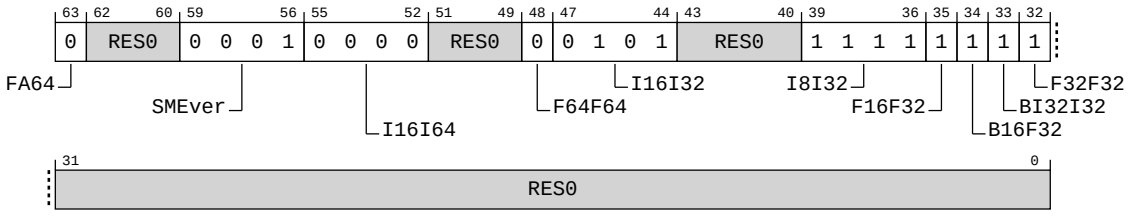


Table A-233: ID_AA64SMFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	FA64	Indicates support at each Exception Level for execution of the full AArch64 Advanced SIMD and SVE instruction sets when the PE is in Streaming SVE mode. **0b0** Only those AArch64 instructions defined as being legal can be executed in streaming SVE mode.	0b0
[62:60]	**RES0**	Reserved	**RES0**
[59:56]	SMEver	Indicates support for SME instructions when FEAT_SME is implemented. **0b0001** As 0b0000, and adds the mandatory SME2 instructions.	0b0001
[55:52]	I16I64	Indicates SME support for the following instructions that accumulate into 64-bit integer elements in the ZA array: • The variants of the ADDHA, ADDVA, SMOPA, SMOPS, SUMOPA, SUMOPS, UMOPA, UMOPS, USMOPA, and USMOPS instructions that accumulate into 64-bit integer tiles. • When FEAT_SME2 is implemented, the variants of the ADD, ADDA, SDOT, SMLALL, SMLSLL, SUB, SUBA, SVDOT, UDOT, UMLALL, UMLSLL, and UVDOT instructions that accumulate into 64-bit integer elements in ZA array vectors. **0b0000** The specified instructions are not implemented.	0b0000
[51:49]	**RES0**	Reserved	**RES0**
[48]	F64F64	Indicates SME support for the following instructions that accumulate into FP64 double-precision floating-point elements in the ZA array: • The variants of the FMOPA and FMOPS instructions that accumulate into double-precision tiles. • When FEAT_SME2 is implemented, the variants of the FADD, FMLA, FMLS, and FSUB instructions that accumulate into double-precision elements in ZA array vectors. **0b0** The specified instructions are not implemented.	0b0
[47:44]	I16I32	Indicates SME2 support for SMOPA (2-way), SMOPS (2-way), UMOPA (2-way), and UMOPS (2-way) instructions that accumulate 16-bit outer products into 32-bit integer tiles. **0b0101** The specified instructions are implemented.	0b0101
[43:40]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[39:36]	I8I32	Indicates SME support for SMOPA, SMOPS, SUMOPA, SUMOPS, UMOPA, UMOPS, USMOPA, and USMOPS instructions that accumulate 8-bit outer products into 32-bit tiles **0b1111** The specified instructions are implemented.	0b1111
[35]	F16F32	Indicates SME support for instructions that accumulate FP16 half-precision floating-point outer products into FP32 single-precision floating-point tiles. **0b1** The FMOPA and FMOPS instructions that accumulate half-precision outer products into single-precision tiles are implemented.	0b1
[34]	B16F32	Indicates SME support for instructions that accumulate BFloat16 outer products into FP32 single-precision floating-point tiles. **0b1** The BFMOPA and BFMOPS instructions that accumulate BFloat16 outer products into single-precision tiles are implemented.	0b1
[33]	BI32I32	Indicates SME support for instructions that accumulate thirty-two 1-bit binary outer products into 32-bit integer tiles. **0b1** The BMOPA and BMOPS instructions that accumulate 1-bit binary outer products into 32-bit integer tiles are implemented.	0b1
[32]	F32F32	Indicates SME support for instructions that accumulate FP32 single-precision floating-point outer products into single-precision floating-point tiles. **0b1** The FMOPA and FMOPS instructions that accumulate single-precision outer products into single-precision tiles are implemented.	0b1
[31:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, ID_AA64SMFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b101

Accessibility

This register is read-only and can be accessed from EL1 and higher.

This register is only accessible from the AArch64 state.
MRS <Xt>, ID_AA64SMFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64SMFR0_EL1;
elsif PSTATE.EL == EL2 then
```

```
    X[t, 64] = ID_AA64SMFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64SMFR0_EL1;
```

A.5.9 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0

Provides top level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

The external register EDDFR gives information from this register.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0001 0000 0000 1111 0001 0001 1111 0100 0001 0000 0011 0000 0101 1000 0001 1010
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-83: AARCH64_ID_AA64DFR0_EL1 bit assignments

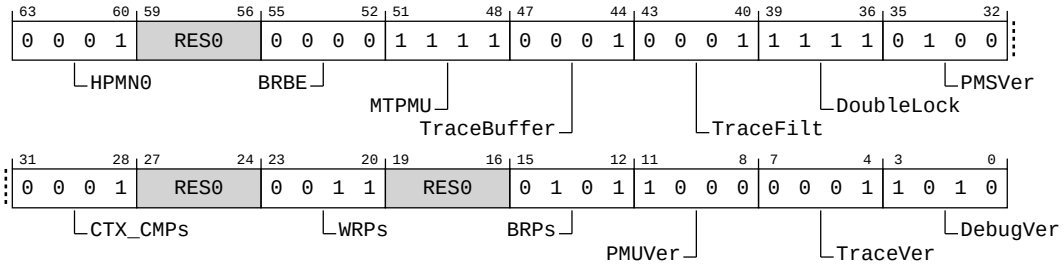


Table A-235: ID_AA64DFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	HPMN0	Zero PMU event counters for a Guest operating system. **0b0001** Setting MDCR_EL2.HPMN to zero has defined behavior.	0b0001

Bits	Name	Description	Reset
[59:56]	**RES0**	Reserved	**RES0**
[55:52]	BRBE	Branch Record Buffer Extension. **0b0000** Branch Record Buffer Extension not implemented.	0b0000
[51:48]	MTPMU	Multi-threaded PMU extension. **0b1111** FEAT_MTPMU not implemented. If FEAT_PMUv3 is implemented, PMEVTYPER<n>_EL0.MT and PMEVTYPER<n>.MT are **RES0**.	0b1111
[47:44]	TraceBuffer	Trace Buffer Extension. **0b0001** Trace Buffer Extension implemented.	0b0001
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. **0b0001** Armv8.4 Self-hosted Trace Extension implemented.	0b0001
[39:36]	DoubleLock	OS Double Lock implemented. **0b1111** OS Double Lock not implemented. OSDLR_EL1 is **RAZ/WI**.	0b1111
[35:32]	PMSVer	Statistical Profiling Extension version. **0b0100** As 0b0011, and adds: • If FEAT_MOPS is implemented, Operation Type packet encodings for Memory Copy and Set operations. • If FEAT_MTE is implemented, Operation Type packet encodings for loads and stores of Allocation Tags.	0b0100
[31:28]	CTX_CMPs	Number of context-aware breakpoints, minus 1. **0b0001** Two context-aware breakpoints are included	0b0001
[27:24]	**RES0**	Reserved	**RES0**
[23:20]	WRPs	Number of watchpoints, minus 1. **0b0011** Four watchpoints	0b0011
[19:16]	**RES0**	Reserved	**RES0**
[15:12]	BRPs	Number of breakpoints, minus 1. **0b0101** Six breakpoints	0b0101

Bits	Name	Description	Reset
[11:8]	PMUVer	Performance Monitors Extension version. This field does not follow the standard ID scheme, but uses the alternative ID scheme described in *Alternative ID scheme used for the Performance Monitors Extension version* in the *Arm® Architecture Reference Manual for A-profile architecture*. **0b1000** PMUv3 for Armv8.8. As 0b0111, and: • Extends the Common event number space to include 0x0040 to 0x00BF and 0x4040 to 0x40BF. • Removes the **CONSTRAINED UNPREDICTABLE** behaviors if a reserved or unimplemented PMU event number is selected.	0b1000
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a trace unit is implemented. **0b0001** Trace unit System registers implemented.	0b0001
[3:0]	DebugVer	Debug architecture version. Indicates presence of Armv8 debug architecture. **0b1010** Armv8.8 debug architecture, FEAT_Debugv8p8.	0b1010

Access

```
MRS <Xt>, ID_AA64DFR0_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0101	0b000

Accessibility

```
MRS <Xt>, ID_AA64DFR0_EL1
```

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64DFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64DFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64DFR0_EL1;
```

A.5.10 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1

Provides top level information about the debug system in AArch64.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0000 0001 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 A-84: AARCH64_ID_AA64DFR1_EL1 bit assignments

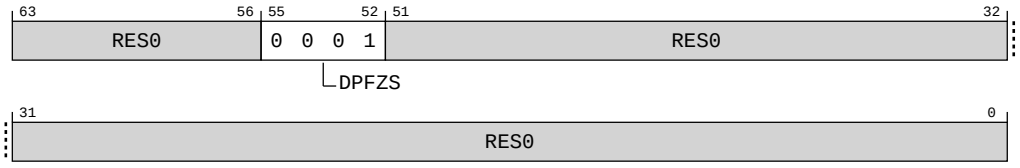


Table A-237: ID_AA64DFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[55:52]	DPFZS	Behavior of the cycle counter when event counting is frozen by a Statistical Profiling management event. **0b0001** The cycle counter PMCCNTR_EL0 does not count when PMCR_EL0.DP is 1 and counting by event counters accessible to EL1 is frozen by the PMCR_EL0.FZS mechanism.	0b0001
[51:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, ID_AA64DFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0101	0b001

Accessibility

MRS <Xt>, ID_AA64DFR1_EL1

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

```
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64DFR1_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64DFR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64DFR1_EL1;
```

A.5.11 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0

Provides information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

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
```

Bit descriptions

Figure A-85: AARCH64_ID_AA64AFR0_EL1 bit assignments

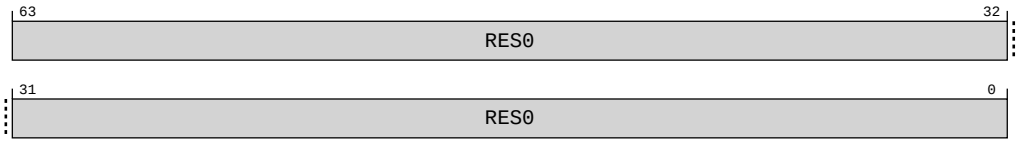


Table A-239: ID_AA64AFR0_EL1 bit descriptions

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

Access

MRS <Xt>, ID_AA64AFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0101	0b100

Accessibility

MRS <Xt>, ID_AA64AFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64AFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64AFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64AFR0_EL1;
```

A.5.12 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1

Reserved for future expansion of information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

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
```

Bit descriptions

Figure A-86: AARCH64_ID_AA64AFR1_EL1 bit assignments

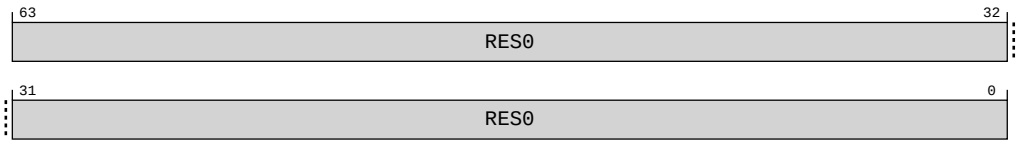


Table A-241: ID_AA64AFR1_EL1 bit descriptions

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

Access

`MRS <Xt>, ID_AA64AFR1_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0101	0b101

Accessibility

`MRS <Xt>, ID_AA64AFR1_EL1`

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64AFR1_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64AFR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64AFR1_EL1;
```

A.5.13 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0

Provides information about the instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0010 0010 0001 0001 xxxx xxxx xxxx 0001 0000 0010 0001 xxxx xxxx xxxx 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 A-87: AARCH64_ID_AA64ISAR0_EL1 bit assignments

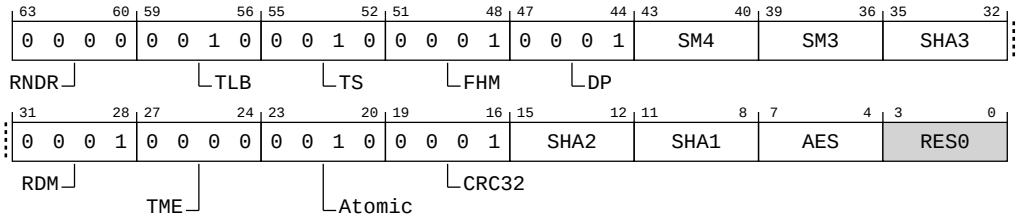


Table A-243: ID_AA64ISAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	RNDR	Indicates support for Random Number instructions in AArch64 state. When FEAT_RNG_TRAP is implemented, the value returned by a direct read of ID_AA64ISAR0_EL1.RNDR is further controlled by the value of SCR_EL3.TRNDR. **0b0000** No Random Number instructions are implemented.	0b0000
[59:56]	TLB	Indicates support for Outer Shareable and TLB range maintenance instructions. **0b0010** Outer Shareable and TLB range maintenance instructions are implemented.	0b0010

Bits	Name	Description	Reset
[55:52]	TS	Indicates support for flag manipulation instructions. **0b0010** CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented.	0b0010
[51:48]	FHM	Indicates support for FMLAL and FMLSL instructions. **0b0001** FMLAL and FMLSL instructions are implemented.	0b0001
[47:44]	DP	Indicates support for Dot Product instructions in AArch64 state. **0b0001** UDOT and SDOT instructions implemented.	0b0001
[43:40]	SM4	Indicates support for SM4 instructions in AArch64 state. **0b0001** SM4E and SM4EKEY instructions implemented. This value applies when CRYPTO == TRUE. **0b0000** No SM4 instructions implemented. This value applies when CRYPTO == FALSE.	The reset values can be the following: 0b0001, 0b0000, respective to the value.
[39:36]	SM3	Indicates support for the following SM3 instructions SM3SS1, SM3TT1A, SM3TT1B, SM3TT2A, SM3TT2B, SM3PARTW1, and SM3PARTW2 in AArch64 state. **0b0001** The specified instructions are implemented. This value applies when CRYPTO == TRUE. **0b0000** The specified instructions are not implemented. This value applies when CRYPTO == FALSE.	The reset values can be the following: 0b0001, 0b0000, respective to the value.
[35:32]	SHA3	Indicates support for the following SHA3 instructions EOR3, RAX1, XAR, and BCAX in AArch64 state. **0b0001** The specified instructions are implemented. This value applies when CRYPTO == TRUE. **0b0000** The specified instructions are not implemented. This value applies when CRYPTO == FALSE.	The reset values can be the following: 0b0001, 0b0000, respective to the value.
[31:28]	RDM	Indicates support for SQRDMLAH and SQRDMLSH instructions in AArch64 state. **0b0001** SQRDMLAH and SQRDMLSH instructions implemented.	0b0001

Bits	Name	Description	Reset
[27:24]	TME	Indicates support for the following TME instructions `TCANCEL`, `TCOMMIT`, `TSTART`, and `TTEST`. **`0b0000`** The specified instructions are not implemented. Access to this field is: RO	`0b0000`
[23:20]	Atomic	Indicates support for Atomic instructions in AArch64 state. **`0b0010`** `LDADD`, `LDCLR`, `LDEOR`, `LDSET`, `LDSMAX`, `LDSMIN`, `LDUMAX`, `LDUMIN`, `CAS`, `CASP`, and `SWP` instructions implemented.	`0b0010`
[19:16]	CRC32	Indicates support for the following CRC32 instructions `CRC32B`, `CRC32H`, `CRC32W`, `CRC32X`, `CRC32CB`, `CRC32CH`, `CRC32CW`, and `CRC32CX` in AArch64 state. **`0b0001`** The specified instructions are implemented.	`0b0001`
[15:12]	SHA2	Indicates support for SHA2 instructions in AArch64 state. **`0b0010`** Implements instructions: • `SHA256H`, `SHA256H2`, `SHA256SU0`, and `SHA256SU1`. • `SHA512H`, `SHA512H2`, `SHA512SU0`, and `SHA512SU1`. This value applies when CRYPTO == TRUE. **`0b0000`** No SHA2 instructions implemented. This value applies when CRYPTO == FALSE.	The reset values can be the following: `0b0010`, `0b0000`, respective to the value.
[11:8]	SHA1	Indicates support for the following SHA1 instructions `SHA1C`, `SHA1P`, `SHA1M`, `SHA1H`, `SHA1SU0`, and `SHA1SU1` in AArch64 state. **`0b0001`** The specified instructions are implemented. This value applies when CRYPTO == TRUE. **`0b0000`** The specified instructions are not implemented. This value applies when CRYPTO == FALSE.	The reset values can be the following: `0b0001`, `0b0000`, respective to the value.
[7:4]	AES	Indicates support for AES instructions in AArch64 state. **`0b0010`** As for `0b0001`, plus `PMULL` and `PMULL2` instructions operating on 64-bit source elements. This value applies when CRYPTO == TRUE. **`0b0000`** No AES instructions implemented. This value applies when CRYPTO == FALSE.	The reset values can be the following: `0b0010`, `0b0000`, respective to the value.

Bits	Name	Description	Reset
[3:0]	**RES0**	Reserved	**RES0**

Access

```
MRS <Xt>, ID_AA64ISAR0_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b000

Accessibility

```
MRS <Xt>, ID_AA64ISAR0_EL1
```

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64ISAR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ISAR0_EL1;
```

A.5.14 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0001 0001 0001 0001 0010 0001 0001 0000 0000 0011 0001 0001 0000 0000 0010
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
```

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

Bit descriptions

Figure A-88: AARCH64_ID_AA64ISAR1_EL1 bit assignments

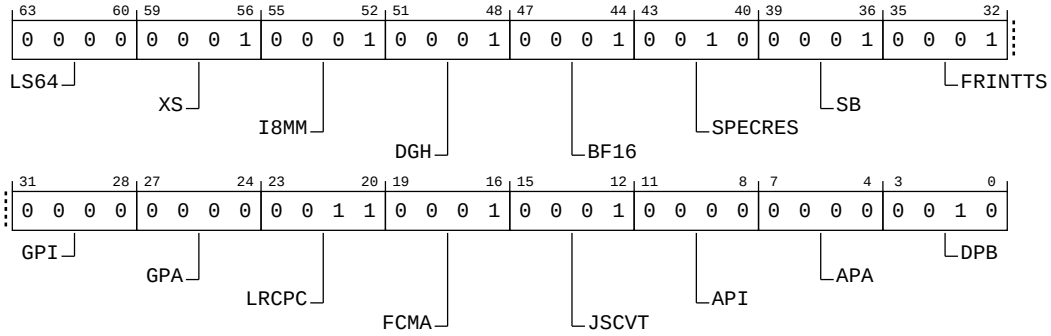


Table A-245: ID_AA64ISAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	LS64	Indicates support for `LD64B` and `ST64B*` instructions, and the ACCDATA_EL1 register. **0b0000** No `LD64B` or `ST64B` instructions are supported.	0b0000
[59:56]	XS	Indicates support for the XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the HCRX_EL2.{FGTnXS, FnXS} fields in AArch64 state. **0b0001** The XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the HCRX_EL2.{FGTnXS, FnXS} fields are supported.	0b0001
[55:52]	I8MM	Indicates support for the following Advanced SIMD Int8 matrix multiplication instructions `SMMLA`, `SUDOT`, `UMMLA`, `USMMLA`, and `USDOT` in AArch64 state. **0b0001** The specified instructions are implemented.	0b0001
[51:48]	DGH	Indicates support for the Data Gathering Hint instruction. **0b0001** Data Gathering Hint is implemented.	0b0001
[47:44]	BF16	Indicates support for Advanced SIMD and floating-point BFloat16 instructions in AArch64 state. **0b0001** `BFCVT`, `BFCVTN`, `BFCVTN2`, `BFDOT`, `BFMLALB`, `BFMLALT`, and `BFMMLA` instructions are implemented.	0b0001
[43:40]	SPECRES	Indicates support for prediction invalidation instructions in AArch64 state. **0b0010** As `0b0001`, and `COSP RCTX` instruction is implemented.	0b0010
[39:36]	SB	Indicates support for `SB` instruction in AArch64 state. **0b0001** `SB` instruction is implemented.	0b0001

Bits	Name	Description	Reset
[35:32]	FRINTTS	Indicates support for FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions. **0b0001** The specified instructions are implemented.	0b0001
[31:28]	GPI	Indicates support for an **IMPLEMENTATION DEFINED** algorithm is implemented in the PE for generic code authentication in AArch64 state. **0b0000** Generic Authentication using an **IMPLEMENTATION DEFINED** algorithm is not implemented.	0b0000
[27:24]	GPA	Indicates whether the QARMA5 algorithm is implemented in the PE for generic code authentication in AArch64 state. **0b0000** Generic Authentication using the QARMA5 algorithm is not implemented.	0b0000
[23:20]	LRCPC	Indicates support for weaker release consistency, RCpc, based model. **0b0011** As 0b0010, and the post-index LDAPR, LDIAPP, STILP, and pre-index STLR instructions are implemented. If Advanced SIMD and floating-point is implemented, then the LDAPUR (SIMD&FP), LDAP1 (SIMD&FP), STLUR (SIMD&FP), and STL1 (SIMD&FP) instructions are implemented in Advanced SIMD and floating-point.	0b0011
[19:16]	FCMA	Indicates support for complex number addition and multiplication, where numbers are stored in vectors. **0b0001** The FCMLA and FCADD instructions are implemented.	0b0001
[15:12]	JSCVT	Indicates support for JavaScript conversion from double precision floating-point values to integers in AArch64 state. **0b0001** The FJCVTZS instruction is implemented.	0b0001
[11:8]	API	Indicates whether an **IMPLEMENTATION DEFINED** algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. **0b0000** Address Authentication using an **IMPLEMENTATION DEFINED** algorithm is not implemented.	0b0000
[7:4]	APA	Indicates whether the QARMA5 algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. **0b0000** Address Authentication using the QARMA5 algorithm is not implemented.	0b0000
[3:0]	DPB	Data Persistence writeback. Indicates support for the DC CVAP and DC CVADP instructions in AArch64 state. **0b0010** DC CVAP and DC CVADP supported.	0b0010

Access

MRS <Xt>, ID_AA64ISAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b001

Accessibility

`MRS <Xt>, ID_AA64ISAR1_EL1`

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64ISAR1_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ISAR1_EL1;
```

A.5.15 ID_AA64ISAR2_EL1, AArch64 Instruction Set Attribute Register 2

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations


Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, **RES0** from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0000 0000 0001 0000 0000 0000 0000 0001 0001 0001 0001 0101 0001 0000 0010
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure A-89: AARCH64_ID_AA64ISAR2_EL1 bit assignments

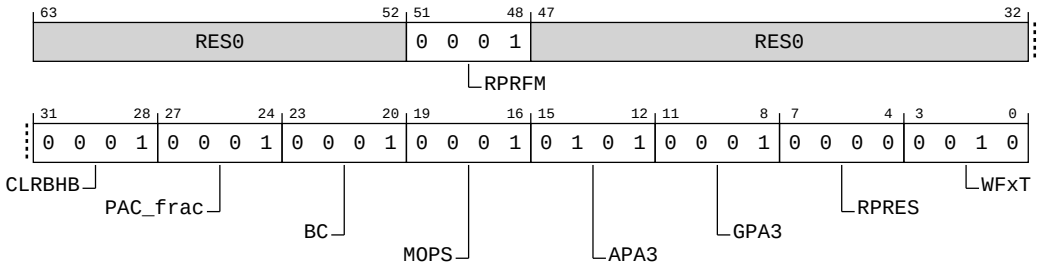


Table A-247: ID_AA64ISAR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:52]	**RES0**	Reserved	**RES0**
[51:48]	RPRFM	RPRFM hint instruction. **0b0001** `RPRFM` hint instruction is implemented.	0b0001
[47:32]	**RES0**	Reserved	**RES0**
[31:28]	CLRBHB	Indicates support for the `CLRBHB` instruction in AArch64 state. **0b0001** `CLRBHB` instruction is implemented.	0b0001
[27:24]	PAC_frac	Indicates which address bit is used to determine the size of the PAC field. **0b0001** The address bit which is used to define the size of the PAC field is fixed.	0b0001
[23:20]	BC	Indicates support for the `BC` instruction in AArch64 state. **0b0001** `BC` instruction is implemented.	0b0001
[19:16]	MOPS	Indicates support for the Memory Copy and Memory Set instructions in AArch64 state. **0b0001** The Memory Copy and Memory Set instructions are implemented in AArch64 state with the following exception. If FEAT_MTE is implemented, then `SETGP*`, `SETGM*` and `SETGE*` instructions are also supported.	0b0001
[15:12]	APA3	Indicates whether the QARMA3 algorithm is implemented in the PE for address authentication in AArch64 state. This applies to all Pointer Authentication instructions other than the `PACGA` instruction. **0b0101** Address Authentication using the QARMA3 algorithm is implemented, FEAT_EPAC is not implemented, FEAT_PAuth2, FEAT_FPAC, and FEAT_FPACCOMBINE are implemented.	0b0101
[11:8]	GPA3	Indicates whether the QARMA3 algorithm is implemented in the PE for generic code authentication in AArch64 state. **0b0001** Generic Authentication using the QARMA3 algorithm is implemented. This includes the `PACGA` instruction.	0b0001

Bits	Name	Description	Reset
[7:4]	RPRES	Indicates support for 12 bits of mantissa in reciprocal and reciprocal square root instructions in AArch64 state, when FPCR.AH is 1. **0b0000** Reciprocal and reciprocal square root estimates give 8 bits of mantissa, when FPCR.AH is 1.	0b0000
[3:0]	WFxT	Indicates support for the `WFET` and `WFIT` instructions in AArch64 state. **0b0010** `WFET` and `WFIT` are supported, and the register number is reported in the ESR_ELx on exceptions.	0b0010

Access

`MRS <Xt>, ID_AA64ISAR2_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b010

Accessibility

`MRS <Xt>, ID_AA64ISAR2_EL1`

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64ISAR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ISAR2_EL1;
```

A.5.16 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

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

Bit descriptions

Figure A-90: AARCH64_ID_AA64MMFR0_EL1 bit assignments

Table A-249: ID_AA64MMFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	ECV	Indicates presence of Enhanced Counter Virtualization. **0b0010** As 0b0001, and also includes support for CNTHCTL_EL2.ECV and CNTPOFF_EL2.	0b0010
[59:56]	FGT	Indicates presence of the Fine-Grained Trap controls. **0b0001** Fine-grained trap controls are implemented. Supports: • If EL2 is implemented, the HAFGRTR_EL2, HDFGRTR_EL2, HDFGWTR_EL2, HFGRTR_EL2, HFGITR_EL2 and HFGWTR_EL2 registers, and their associated traps. • If EL2 is implemented, MDCR_EL2.TDCC. • If EL3 is implemented, MDCR_EL3.TDCC. • If both EL2 and EL3 are implemented, SCR_EL3.FGTEn.	0b0001
[55:48]	**RES0**	Reserved	**RES0**
[47:44]	ExS	Indicates support for disabling context synchronizing exception entry and exit. **0b0000** All exception entries and exits are context synchronization events.	0b0000
[43:40]	TGran4_2	Indicates support for 4KB memory granule size at stage 2. **0b0010** 4KB granule supported at stage 2.	0b0010
[39:36]	TGran64_2	Indicates support for 64KB memory granule size at stage 2. **0b0010** 64KB granule supported at stage 2.	0b0010

Bits	Name	Description	Reset
[35:32]	TGran16_2	Indicates support for 16KB memory granule size at stage 2. **0b0010** 16KB granule supported at stage 2.	0b0010
[31:28]	TGran4	Indicates support for 4KB memory translation granule size. **0b0000** 4KB granule supported.	0b0000
[27:24]	TGran64	Indicates support for 64KB memory translation granule size. **0b0000** 64KB granule supported.	0b0000
[23:20]	TGran16	Indicates support for 16KB memory translation granule size. **0b0001** 16KB granule supported.	0b0001
[19:16]	**RES0**	Reserved	**RES0**
[15:12]	SNSMem	Indicates support for a distinction between Secure and Non-secure Memory. **0b0001** Does support a distinction between Secure and Non-secure Memory.	0b0001
[11:8]	BigEnd	Indicates support for mixed-endian configuration. **0b0001** Mixed-endian support. The SCTLR_ELx.EE and SCTLR_EL1.E0E bits can be configured.	0b0001
[7:4]	ASIDBits	Number of ASID bits. **0b0010** 16 bits.	0b0010
[3:0]	PARange	Physical Address range supported. **0b0010** 40 bits, 1TB.	0b0010

Access

`MRS <Xt>, ID_AA64MMFR0_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b000

Accessibility

`MRS <Xt>, ID_AA64MMFR0_EL1`

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR0_EL1;
elsif PSTATE.EL == EL2 then
```

```
    X[t, 64] = ID_AA64MMFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64MMFR0_EL1;
```

A.5.17 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0001 0001 0001 0001 0001 0001 0010 0000 0001 0000 0011 0001 0010 0001 0010 0011
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-91: AARCH64_ID_AA64MMFR1_EL1 bit assignments

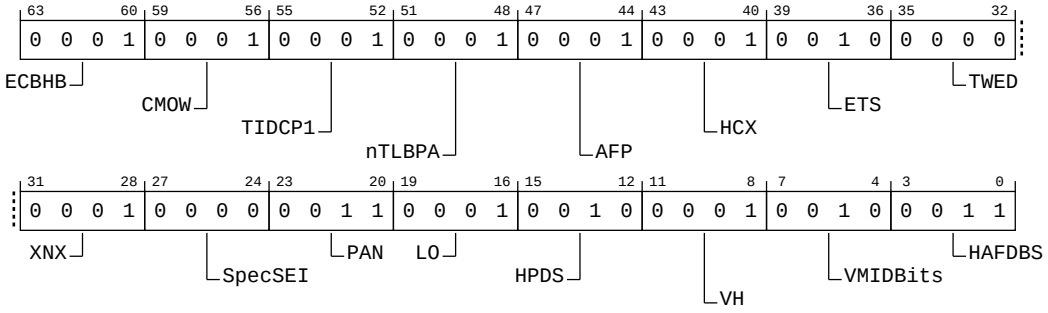


Table A-251: ID_AA64MMFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	ECBHB	Indicates support for restrictions on branch history speculation around exceptions. **0b0001** The branch history information created in a context before an exception to a higher Exception level using AArch64 cannot be used by code before that exception to exploitatively control the execution of any indirect branches in code in a different context after the exception.	0b0001
[59:56]	CMOW	Indicates support for cache maintenance instruction permission. **0b0001** SCTLR_EL1.CMOW is implemented. If EL2 is implemented, SCTLR_EL2.CMOW and HCRX_EL2.CMOW bits are implemented.	0b0001
[55:52]	TIDCP1	Indicates whether SCTLR_EL1.TIDCP and SCTLR_EL2.TIDCP are implemented in AArch64 state. **0b0001** SCTLR_EL1.TIDCP bit is implemented. If EL2 is implemented, SCTLR_EL2.TIDCP bit is implemented.	0b0001
[51:48]	nTLBPA	Indicates support for intermediate caching of translation table walks. **0b0001** The intermediate caching of translation table walks does not include non-coherent physical translation caches.	0b0001
[47:44]	AFP	Indicates support for FPCR.{AH, FIZ, NEP}. **0b0001** The FPCR.{AH, FIZ, NEP} fields are supported.	0b0001
[43:40]	HCX	Indicates support for HCRX_EL2 and its associated EL3 trap. **0b0001** HCRX_EL2 and its associated EL3 trap are supported.	0b0001
[39:36]	ETS	Indicates support for Enhanced Translation Synchronization. **0b0010** Enhanced Translation Synchronization is supported.	0b0010
[35:32]	TWED	Indicates support for the configurable delayed trapping of WFE. **0b0000** Configurable delayed trapping of WFE is not supported.	0b0000
[31:28]	XNX	Indicates support for execute-never control distinction by Exception level at stage 2. **0b0001** Distinction between EL0 and EL1 execute-never control at stage 2 supported.	0b0001
[27:24]	SpecSEI	Describes whether the PE can generate SError exceptions from speculative reads of memory, including speculative instruction fetches. **0b0000** The PE never generates an SError exception due to an External abort on a speculative read.	0b0000
[23:20]	PAN	Privileged Access Never. Indicates support for the PAN bit in PSTATE, SPSR_EL1, SPSR_EL2, SPSR_EL3, and DSPSR_EL0. **0b0011** PAN supported, `AT S1E1RP` and `AT S1E1WP` instructions supported, and SCTLR_EL1.EPAN and SCTLR_EL2.EPAN bits supported.	0b0011

Bits	Name	Description	Reset
[19:16]	LO	LORegions. Indicates support for LORegions. **0b0001** LORegions supported.	0b0001
[15:12]	HPDS	Hierarchical Permission Disables. Indicates support for disabling hierarchical controls in translation tables. **0b0010** As for value 0b0001, and adds possible hardware allocation of bits[62:59] of the Translation table descriptors from the final lookup level for **IMPLEMENTATION DEFINED** use.	0b0010
[11:8]	VH	Virtualization Host Extensions. **0b0001** Virtualization Host Extensions supported.	0b0001
[7:4]	VMIDBits	Number of VMID bits. **0b0010** 16 bits	0b0010
[3:0]	HAFDBS	Hardware updates to Access flag and Dirty state in translation tables. **0b0011** As 0b0010, and adds support for hardware update of the Access flag for Table descriptors.	0b0011

Access

MRS <Xt>, ID_AA64MMFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b001

Accessibility

MRS <Xt>, ID_AA64MMFR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR1_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64MMFR1_EL1;
```

A.5.18 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, **RES0** from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0001 0010 xxxx 0001 0000 0001 0001 0001 0001 0000 0001 0000 0001 0000 0001 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-92: AARCH64_ID_AA64MMFR2_EL1 bit assignments

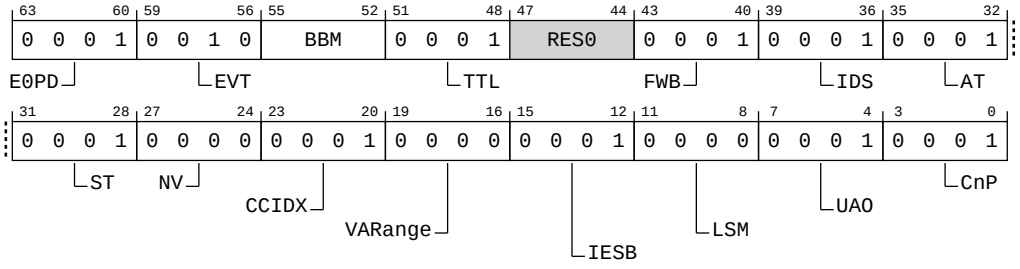


Table A-253: ID_AA64MMFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	E0PD	Indicates support for the E0PD mechanism. **0b0001** E0PDx mechanism is implemented.	0b0001
[59:56]	EVT	Enhanced Virtualization Traps. If EL2 is implemented, indicates support for the HCR_EL2.{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps. **0b0010** HCR_EL2.{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps are supported.	0b0010
[55:52]	BBM	Allows identification of the requirements of the hardware to have break-before-make sequences when changing block size for a translation. **0b0000** Level 0 support for changing block size is supported. **0b0001** Level 1 support for changing block size is supported. **0b0010** Level 2 support for changing block size is supported.	The reset values can be the following: 0b0000, 0b0001, 0b0010, respective to the value.
[51:48]	TTL	Indicates support for TTL field in address operations. **0b0001** TLB maintenance instructions by address have bits[47:44] holding the TTL field.	0b0001
[47:44]	**RES0**	Reserved	**RES0**
[43:40]	FWB	Indicates support for HCR_EL2.FWB. **0b0001** HCR_EL2.FWB is supported.	0b0001
[39:36]	IDS	Indicates the value of ESR_ELx.EC that reports an exception generated by a read access to the feature ID space. **0b0001** All exceptions generated by an AArch64 read access to the feature ID space are reported by ESR_ELx.EC == 0x18.	0b0001
[35:32]	AT	Identifies support for unaligned single-copy atomicity and atomic functions. **0b0001** Unaligned single-copy atomicity and atomic functions with a 16-byte address range aligned to 16-bytes are supported.	0b0001
[31:28]	ST	Identifies support for small translation tables. **0b0001** The maximum value of the TCR_ELx.{T0SZ,T1SZ} and VTCR_EL2.T0SZ fields is 48 for 4KB and 16KB granules, and 47 for 64KB granules.	0b0001
[27:24]	NV	Nested Virtualization. If EL2 is implemented, indicates support for the use of nested virtualization. **0b0000** Nested virtualization is not supported.	0b0000

Bits	Name	Description	Reset
[23:20]	CCIDX	Support for the use of revised CCSIDR_EL1 register format. **0b0001** 64-bit format implemented for all levels of the CCSIDR_EL1.	0b0001
[19:16]	VARange	Indicates support for a larger virtual address. **0b0000** VMSAv8-64 supports 48-bit VAs.	0b0000
[15:12]	IESB	Indicates support for the IESB bit in the SCTLR_ELx registers. **0b0001** IESB bit in the SCTLR_ELx registers is supported.	0b0001
[11:8]	LSM	Indicates support for LSMAOE and nTLSMD bits in SCTLR_EL1 and SCTLR_EL2. **0b0000** LSMAOE and nTLSMD bits not supported.	0b0000
[7:4]	UAO	User Access Override. **0b0001** UAO supported.	0b0001
[3:0]	CnP	Indicates support for Common not Private translations. **0b0001** Common not Private translations supported.	0b0001

Access

MRS <Xt>, ID_AA64MMFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b010

Accessibility

MRS <Xt>, ID_AA64MMFR2_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64MMFR2_EL1;
```

A.5.19 ID_AA64MMFR3_EL1, AArch64 Memory Model Feature Register 3

Provides information about the implemented memory model and memory management support in AArch64 state.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, **RES0** from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

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

Bit descriptions

Figure A-93: AARCH64_ID_AA64MMFR3_EL1 bit assignments

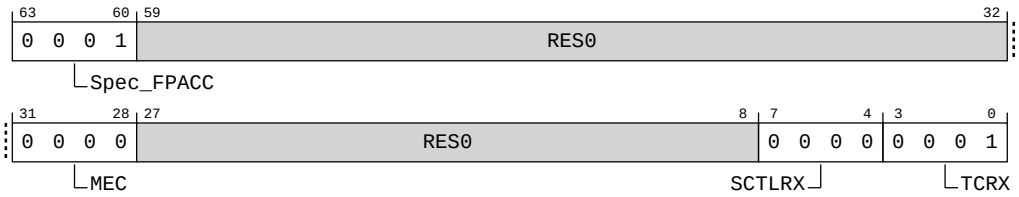


Table A-255: ID_AA64MMFR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	Spec_FPACC	Speculative behavior in the event of a PAC authentication failure in an implementation that includes FEAT_FPACCOMBINE. **0b0001** The speculative use of pointers processed by a PAC Authentication is not materially different in terms of the impact on cached microarchitectural state between passing and failing of the PAC Authentication.	0b0001
[59:32]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[31:28]	MEC	Indicates support for Memory Encryption Contexts. **0b0000** Memory Encryption Contexts is not supported.	0b0000
[27:8]	**RES0**	Reserved	**RES0**
[7:4]	SCTLRX	SCTLR Extension. Indicates support for extension of SCTLR_ELx. **0b0000** SCTLR2_EL1, SCTLR2_EL2, SCTLR2_EL3 registers, and their associated trap controls are not implemented.	0b0000
[3:0]	TCRX	TCR Extension. Indicates support for extension of TCR_ELx. **0b0001** TCR2_EL1, TCR2_EL2, and their associated trap controls are implemented.	0b0001

Access

`MRS <Xt>, ID_AA64MMFR3_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b011

Accessibility

`MRS <Xt>, ID_AA64MMFR3_EL1`

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR3_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR3_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64MMFR3_EL1;
```

A.5.20 MPAMIDR_EL1, MPAM ID Register (EL1)

Indicates the presence and maximum PARTID and PMG values supported in the implementation. It also indicates whether the implementation supports MPAM virtualization.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0010 0100 0000 0000 0000 0000 0000 0001 0000 0000 0000 0110 0000 0000 0011 1111
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

MPAMIDR_EL1 indicates the MPAM implementation parameters of the PE.

Figure A-94: AARCH64_MPAMIDR_EL1 bit assignments

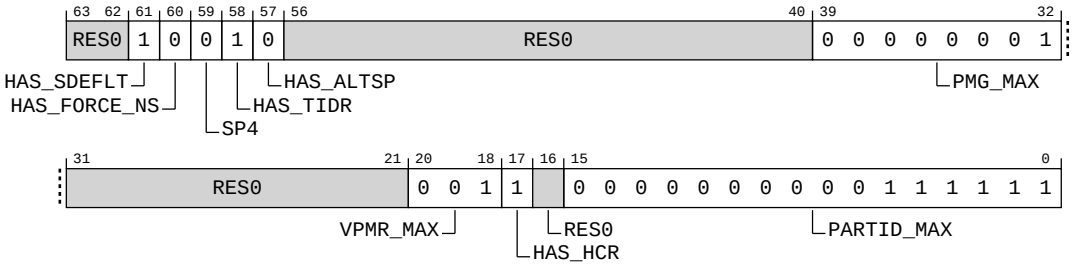


Table A-257: MPAMIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:62]	**RES0**	Reserved	**RES0**
[61]	HAS_SDEFLT	HAS_SDEFLT indicates support for MPAM3_EL3.SDEFLT bit. **0b1** The SDEFLT bit is implemented in MPAM3_EL3.	0b1
[60]	HAS_FORCE_NS	HAS_FORCE_NS indicates support for MPAM3_EL3.FORCE_NS bit. **0b0** The FORCE_NS bit is not implemented in MPAM3_EL3.	0b0
[59]	SP4	Supports 4 MPAM PARTID spaces. **0b0** MPAM supports 2 PARTID spaces.	0b0
[58]	HAS_TIDR	HAS_TIDR indicates support for MPAM2_EL2.TIDR bit. **0b1** The TIDR bit is implemented in MPAM2_EL2.	0b1
[57]	HAS_ALTSP	HAS_ALTSP indicates support for alternative PARTID spaces. **0b0** Alternative PARTID spaces are not implemented.	0b0
[56:40]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[39:32]	PMG_MAX	The largest value of PMG that the implementation can generate. The PMG_I and PMG_D fields of every MPAMn_ELx must implement at least enough bits to represent PMG_MAX. **0x01** Max PMG field is 1	0x01
[31:21]	**RES0**	Reserved	**RES0**
[20:18]	VPMR_MAX	Indicates the maximum register index n for the MPAMVPM<n>_EL2 registers. **0b001** Two MPAMVPMn_EL2 registers are implemented	0b001
[17]	HAS_HCR	HAS_HCR indicates that the PE implementation supports MPAM virtualization, including MPAMHCR_EL2, MPAMVPMV_EL2, and MPAMVPM<n>_EL2 with n in the range 0 to VPMR_MAX. Must be 0 if EL2 is not implemented in either Security state. **0b1** MPAM virtualization is supported.	0b1
[16]	**RES0**	Reserved	**RES0**
[15:0]	PARTID_MAX	The largest value of PARTID that the implementation can generate. The PARTID_I and PARTID_D fields of every MPAMn_ELx must implement at least enough bits to represent PARTID_MAX. **0x003F** Max PARTID field is 63	0x003F

Access

MRS <Xt>, MPAMIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b100

Accessibility

MRS <Xt>, MPAMIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && MPAMHCR_EL2.TRAP_MPAMIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MPAM2_EL2.TIDR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = MPAMIDR_EL1;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMIDR_EL1;
```

A.5.21 IMP_CPUCFR_EL1, CPU Configuration Register

This register provides configuration information for the core

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0x10 0x0x
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-95: AARCH64_IMP_CPUCFR_EL1 bit assignments

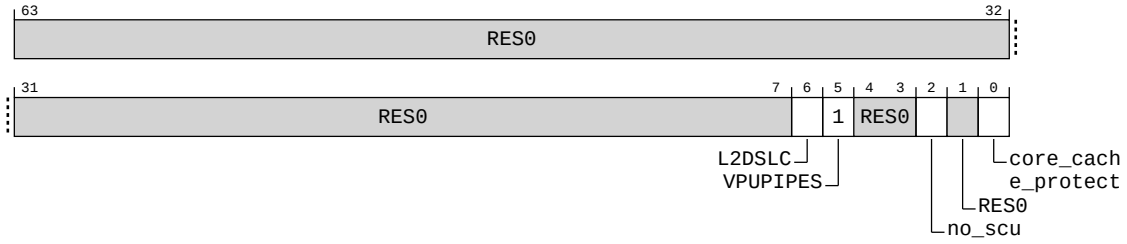


Table A-259: IMP_CPUCFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:7]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[6]	L2DSLC	L2 data RAM register slice presence. Possible values of this bit are: **0b0** No L2 data RAM register slice present **0b1** L2 data RAM register slice present	The reset values can be the following: `0b0`, `0b1`, respective to the value.
[5]	VPUPIPES	Indicates the number of Vector Processing Unit (VPU) pipes. Possible values of this bit are: **0b1** 6 x 128-bit	`0b1`
[4:3]	**RES0**	Reserved	**RES0**
[2]	no_scu	Indicates whether the SCU is present or not. Possible values of this bit are: **0b0** The SCU is present. **0b1** The SCU is not present.	The reset values can be the following: `0b0`, `0b1`, respective to the value.
[1]	**RES0**	Reserved	**RES0**
[0]	core_cache_protect	Indicates whether ECC is present or not. Possible values of this field are: **0b0** ECC is not present. **0b1** ECC is present.	The reset values can be the following: `0b0`, `0b1`, respective to the value.

Access

`MRS <Xt>, S3_0_C15_C0_0`

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0000	0b000

Accessibility

`MRS <Xt>, S3_0_C15_C0_0`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUCFR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUCFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUCFR_EL1;
```

A.5.22 CLIDR_EL1, Cache Level ID Register

Identifies the type of cache, or caches, that are implemented at each level and can be managed using the architected cache maintenance instructions that operate by set/way, up to a maximum of seven levels. Also identifies the Level of Coherence (LoC) and Level of Unification (LoU) for the cache hierarchy.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0xxx xxx0 1100 0011 0000 0000 0000 0001 0010 0011
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure A-96: AARCH64_CLIDR_EL1 bit assignments

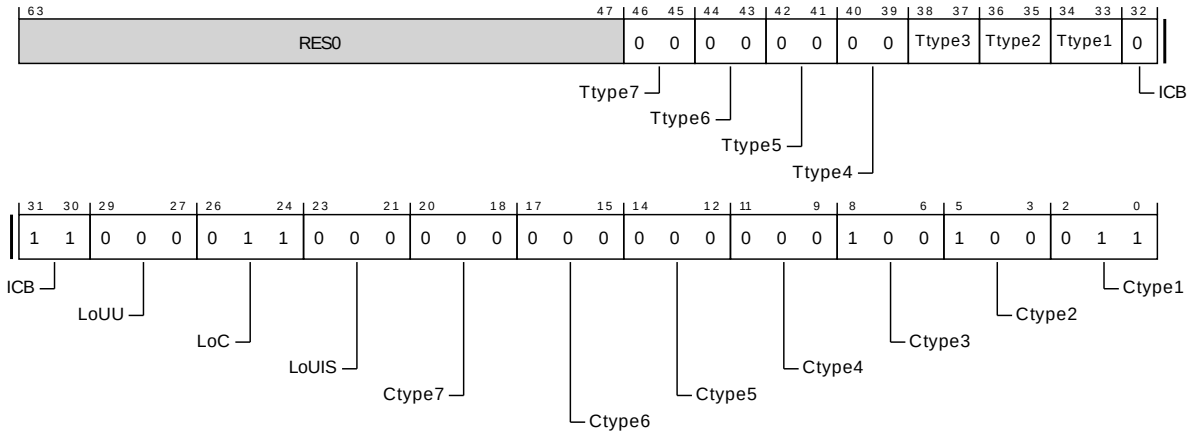


Table A-261: CLIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:47]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[46:45]	Ttype7	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache.	0b00
[44:43]	Ttype6	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache.	0b00
[42:41]	Ttype5	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache.	0b00
[40:39]	Ttype4	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache.	0b00
[38:37]	Ttype3	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache. This value applies when BROADCASTMTE == FALSE. **0b10** Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines. This value applies when BROADCASTMTE == TRUE.	xx
[36:35]	Ttype2	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache. This value applies when BROADCASTMTE == FALSE. **0b10** Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines. This value applies when BROADCASTMTE == TRUE.	xx

Bits	Name	Description	Reset
[34:33]	Ttype1	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache. This value applies when BROADCASTMTE == FALSE. **0b10** Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines. This value applies when BROADCASTMTE == TRUE.	xx
[32:30]	ICB	Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions. **0b011** L3 cache is the highest Inner Cacheable level.	0b011
[29:27]	LoUU	Level of Unification Uniprocessor for the cache hierarchy. For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions. **Note:** This field does not describe the requirements for instruction cache invalidation. See CTR_EL0.DIC. **Note:** When FEAT_S2FWB is implemented, the architecture requires that this field is zero so that no levels of data cache need to be cleaned in order to manage coherency with instruction fetches. **0b000** Level of Unification Uniprocessor is before the L1 data cache.	0b000
[26:24]	LoC	Level of Coherence for the cache hierarchy. For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions. **0b011**	0b011
[23:21]	LoUIS	Level of Unification Inner Shareable for the cache hierarchy. For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions. **Note:** This field does not describe the requirements for instruction cache invalidation. See CTR_EL0.DIC. **Note:** When FEAT_S2FWB is implemented, the architecture requires that this field is zero so that no levels of data cache need to be cleaned in order to manage coherency with instruction fetches. **0b000** No cache level needs cleaning to Point of Unification	0b000

Bits	Name	Description	Reset
[20:18]	Ctype7	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[17:15]	Ctype6	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[14:12]	Ctype5	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[11:9]	Ctype4	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[8:6]	Ctype3	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b100** Unified cache.	0b100
[5:3]	Ctype2	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b100** Unified cache.	0b100
[2:0]	Ctype1	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b011** Separate instruction and data caches.	0b011

Access

MRS <Xt>, CLIDR_EL1

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

Accessibility

MRS <Xt>, CLIDR_EL1

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

```
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.CLIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CLIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CLIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CLIDR_EL1;
```

A.5.23 GMID_EL1, Multiple tag transfer ID Register

Indicates the block size that is accessed by the LDGM and STGM System instructions.

Configurations

This register is present only when IsFeatureImplemented(FEAT_MTE2). Otherwise, direct accesses to GMID_EL1 are UNDEFINED.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

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

Bit descriptions

Figure A-97: AARCH64_GMID_EL1 bit assignments

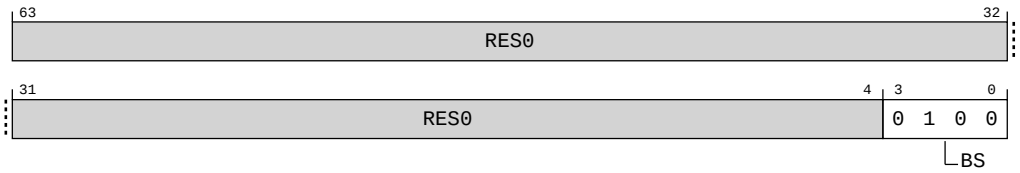


Table A-263: GMID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:4]	**RES0**	Reserved	**RES0**
[3:0]	BS	Log_2 of the block size in words. The minimum supported size is 16B (value == 2) and the maximum is 256B (value == 6). **0b0100** 64 bytes.	0b0100

Access

MRS <Xt>, GMID_EL1

op0	op1	CRn	CRm	op2
0b11	0b001	0b0000	0b0000	0b100

Accessibility

MRS <Xt>, GMID_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID5 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = GMID_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = GMID_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = GMID_EL1;
```

A.5.24 CTR_EL0, Cache Type Register

Provides information about the architecture of the caches.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0100 1001 0100 0100 0100 1100 0000 0000 0100
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-98: AARCH64_CTR_EL0 bit assignments

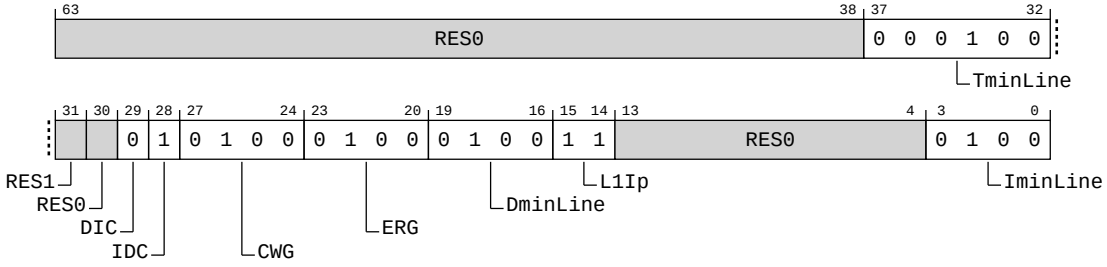


Table A-265: CTR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:38]	**RES0**	Reserved	**RES0**
[37:32]	TminLine	Tag minimum Line. Log_2 of the number of words covered by Allocation Tags in the smallest cache line of all caches which can contain Allocation tags that are controlled by the PE. **0b000100** Log2 of number of words (64/4=16) covered by Allocation Tags in the smallest cache line of all caches	0b000100
[31]	**RES1**	Reserved	**RES1**
[30]	**RES0**	Reserved	**RES0**
[29]	DIC	Instruction cache invalidation requirements for data to instruction coherence. **0b0** Instruction cache invalidation to the Point of Unification is required for data to instruction coherence.	0b0
[28]	IDC	Data cache clean requirements for instruction to data coherence. The meaning of this bit is: **0b1** Data cache clean to the Point of Unification is not required for instruction to data coherence.	0b1

Bits	Name	Description	Reset
[27:24]	CWG	Cache writeback granule. Log_2 of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified. A value of `0b0000` indicates that this register does not provide Cache writeback granule information and either: • The architectural maximum of 512 words (2KB) must be assumed. • The Cache writeback granule can be determined from maximum cache line size encoded in the Cache Size ID Registers. Values greater than `0b1001` are reserved. Arm recommends that an implementation that does not support cache write-back implements this field as `0b0001`. This applies, for example, to an implementation that supports only write-through caches. **0b0100** 64 bytes.	`0b0100`
[23:20]	ERG	Exclusives reservation granule. Log_2 of the number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions. The use of the value `0b0000` is deprecated. The value `0b0001` and values greater than `0b1001` are reserved. **0b0100** 64 bytes.	`0b0100`
[19:16]	DminLine	Log_2 of the number of words in the smallest cache line of all the data caches and unified caches that are controlled by the PE. **0b0100** 64 bytes.	`0b0100`
[15:14]	L1Ip	Level 1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache. **0b11** Physical Index, Physical Tag (PIPT).	`0b11`
[13:4]	**RES0**	Reserved	**RES0**
[3:0]	IminLine	Log_2 of the number of words in the smallest cache line of all the instruction caches that are controlled by the PE. **0b0100** 64 bytes.	`0b0100`

Access

`MRS <Xt>, CTR_EL0`

op0	op1	CRn	CRm	op2
`0b11`	`0b011`	`0b0000`	`0b0000`	`0b001`

Accessibility

`MRS <Xt>, CTR_EL0`

```
if PSTATE.EL == EL0 then
    if !ELIsInHost(EL0) && SCTLR_EL1.UCT == '0' then
```

```
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HFGRTR_EL2.CTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ELIsInHost(EL0) && SCTLR_EL2.UCT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CTR_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.CTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CTR_EL0;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CTR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CTR_EL0;
```

A.5.25 DCZID_EL0, Data Cache Zero ID Register

Indicates the block size that is written with byte values of 0 by the DC ZVA (Data Cache Zero by Address) System instruction.

If FEAT_MTE is implemented, this register also indicates the granularity at which the DC GVA and DC GZVA instructions write.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification Registers

Access type

RO

Reset value

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

Bit descriptions

Figure A-99: AARCH64_DCZID_EL0 bit assignments

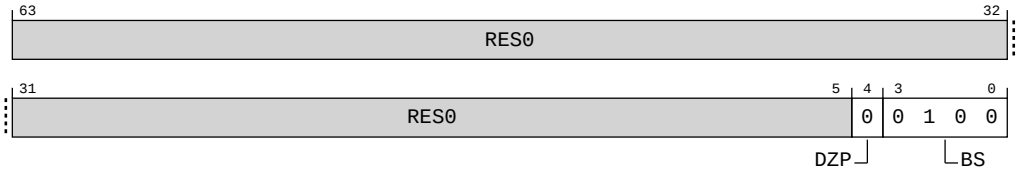


Table A-267: DCZID_EL0 bit descriptions

Bits	Name	Description	Reset
[63:5]	**RES0**	Reserved	**RES0**
[4]	DZP	Data Zero Prohibited. This field indicates whether use of `DC ZVA` instructions is permitted or prohibited. If FEAT_MTE is implemented, this field also indicates whether use of the `DC GVA` and `DC GZVA` instructions are permitted or prohibited. **0b0** Instructions are permitted.	0b0
[3:0]	BS	Log_2 of the block size in words. The maximum size supported is 2KB, indicated by value `0b1001`. If FEAT_MTE2 is implemented, the minimum size supported is 16 bytes, indicated by value `0b0010`. **0b0100** 64 bytes.	0b0100

Access

`MRS <Xt>, DCZID_EL0`

op0	op1	CRn	CRm	op2
0b11	0b011	0b0000	0b0000	0b111

Accessibility

`MRS <Xt>, DCZID_EL0`

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HFGRTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = DCZID_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = DCZID_EL0;
elsif PSTATE.EL == EL2 then
    X[t, 64] = DCZID_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = DCZID_EL0;
```

A.6 AArch64 MPAM registers summary

The following summary table provides an overview of all MPAM registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-269: MPAM registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
MPAM1_EL1	3	0	C10	C5	0	See individual bit resets.	64-bit	MPAM1 Register (EL1)
MPAM0_EL1	3	0	C10	C5	1	See individual bit resets.	64-bit	MPAM0 Register (EL1)
MPAMSM_EL1	3	0	C10	C5	3	See individual bit resets.	64-bit	MPAM Streaming Mode Register
MPAMHCR_EL2	3	4	C10	C4	0	See individual bit resets.	64-bit	MPAM Hypervisor Control Register (EL2)
MPAMVPMV_EL2	3	4	C10	C4	1	See individual bit resets.	64-bit	MPAM Virtual Partition Mapping Valid Register
MPAM2_EL2	3	4	C10	C5	0	See individual bit resets.	64-bit	MPAM2 Register (EL2)
MPAMVPM0_EL2	3	4	C10	C6	0	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 0
MPAMVPM1_EL2	3	4	C10	C6	1	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 1
MPAM3_EL3	3	6	C10	C5	0	See individual bit resets.	64-bit	MPAM3 Register (EL3)

A.6.1 MPAMSM_EL1, MPAM Streaming Mode Register

Holds information to generate MPAM labels for memory requests that are:

- Issued due to the execution of SME load and store instructions.
- Issued when the PE is in Streaming SVE mode due to the execution of SVE and SIMD&FP load and store instructions and SVE prefetch instructions.

If an implementation uses a shared SMCU, then the MPAM labels in this register have precedence over the labels in MPAM0_EL1, MPAM1_EL1, MPAM2_EL2, and MPAM3_EL3.

If an implementation includes an SMCU that is not shared with other PEs, then it is **IMPLEMENTATION DEFINED** whether the MPAM labels in this register have precedence over the labels in MPAM0_EL1, MPAM1_EL1, MPAM2_EL2, and MPAM3_EL3.

The MPAM labels in this register are only used if MPAM1_EL1.MPAMEN is 1.

For memory requests issued from EL0, the MPAM PARTID in this register is virtual and mapped into a physical PARTID when all of the following are true:

- EL2 is implemented and enabled in the current Security state, and the Effective value of HCR_EL2.{E2H, TGE} is not {1, 1}.
- The MPAM virtualization option is implemented and MPAMHCR_EL2.EL0_VPMEN is 1.

For memory requests issued from EL1, the MPAM PARTID in this register is virtual and mapped into a physical PARTID when all of the following are true:

- EL2 is implemented and enabled in the current Security state.
- The MPAM virtualization option is implemented and MPAMHCR_EL2.EL1_VPMEN is 1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

MPAM

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-100: AARCH64_MPAMSM_EL1 bit assignments

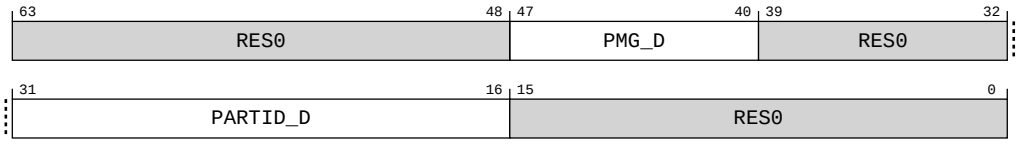


Table A-270: MPAMSM_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:40]	PMG_D	Performance monitoring group property for PARTID_D.	8{x}

Bits	Name	Description	Reset
[39:32]	**RES0**	Reserved	**RES0**
[31:16]	PARTID_D	Partition ID for requests issued due to the execution at any Exception level of SME load and store instructions and, when the PE is in Streaming SVE mode, SVE and SIMD&FP load and store instructions and SVE prefetch instructions.	`16{x}`
[15:0]	**RES0**	Reserved	**RES0**

Access

`MRS <Xt>, MPAMSM_EL1`

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0101	0b011

`MSR MPAMSM_EL1, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0101	0b011

Accessibility

None of the fields in this register are permitted to be cached in a TLB.

`MRS <Xt>, MPAMSM_EL1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && MPAM2_EL2.EnMPAMSM == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = MPAMSM_EL1;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMSM_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMSM_EL1;
```

`MSR MPAMSM_EL1, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && MPAM2_EL2.EnMPAMSM == '0' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        MPAMSM_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMSM_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMSM_EL1 = X[t, 64];
```

A.6.2 MPAMVPMV_EL2, MPAM Virtual Partition Mapping Valid Register

Valid bits for virtual PARTID mapping entries. Each bit m corresponds to virtual PARTID mapping entry m in the MPAMVPM<n>_EL2 registers where n = m >> 2.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

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-101: AARCH64_MPAMVPMV_EL2 bit assignments

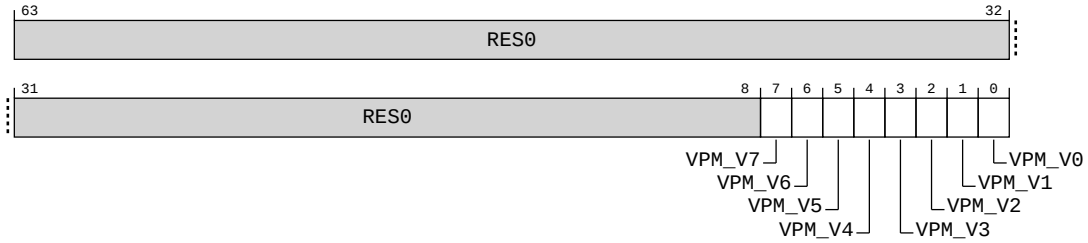


Table A-273: MPAMVPMV_EL2 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:0]	VPM_V<m>, bit[m], where m = 7 to 0	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	8{x}

Access

MRS <Xt>, MPAMVPMV_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0100	0b001

MSR MPAMVPMV_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0100	0b001

Accessibility

MRS <Xt>, MPAMVPMV_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'1x1'} then
        X[t, 64] = NVMem[0x938];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if EL3SDDUndef() then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    else
        X[t, 64] = MPAMVPMV_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPMV_EL2;
```

```
MSR MPAMVPMV_EL2, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'1x1'} then
        NVMem[0x938] = X[t, 64];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if EL3SDDUndef() then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPMV_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMVPMV_EL2 = X[t, 64];
```

A.6.3 MPAMVPM0_EL2, MPAM Virtual PARTID Mapping Register 0

MPAMVPM0_EL2 provides mappings from virtual PARTIDs 0 - 3 to physical PARTIDs.

MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 register. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, MPAMVPM0_EL2.

Virtual PARTID mapping is enabled by MPAMHCR_EL2.EL1_VPMEN for PARTIDs in MPAM1_EL1 and by MPAMHCR_EL2.EL0_VPMEN for PARTIDs in MPAM0_EL1.

A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is valid only when the MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

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-102: AARCH64_MPAMVPM0_EL2 bit assignments

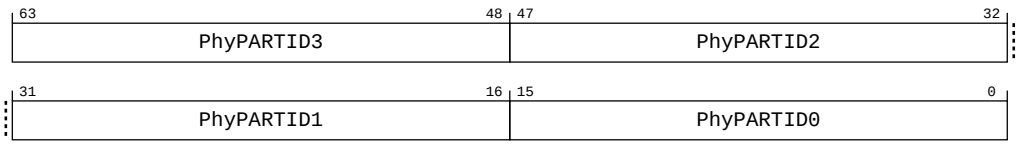


Table A-276: MPAMVPM0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID3	Virtual PARTID Mapping Entry for virtual PARTID 3. PhyPARTID3 gives the mapping of virtual PARTID 3 to a physical PARTID.	16{x}
[47:32]	PhyPARTID2	Virtual PARTID Mapping Entry for virtual PARTID 2. PhyPARTID2 gives the mapping of virtual PARTID 2 to a physical PARTID.	16{x}
[31:16]	PhyPARTID1	Virtual PARTID Mapping Entry for virtual PARTID 1. PhyPARTID1 gives the mapping of virtual PARTID 1 to a physical PARTID.	16{x}
[15:0]	PhyPARTID0	Virtual PARTID Mapping Entry for virtual PARTID 0. PhyPARTID0 gives the mapping of virtual PARTID 0 to a physical PARTID.	16{x}

Access

`MRS <Xt>, MPAMVPM0_EL2`

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b000

`MSR MPAMVPM0_EL2, <Xt>`

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b000

Accessibility

MRS <Xt>, MPAMVPM0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'1x1'} then
        X[t, 64] = NVMem[0x940];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if EL3SDDUndef() then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMVPM0_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPM0_EL2;
```

MSR MPAMVPM0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'1x1'} then
        NVMem[0x940] = X[t, 64];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if EL3SDDUndef() then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM0_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMVPM0_EL2 = X[t, 64];
```

A.6.4 MPAMVPM1_EL2, MPAM Virtual PARTID Mapping Register 1

MPAMVPM1_EL2 provides mappings from virtual PARTIDs 4 - 7 to physical PARTIDs.

MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM0_EL2 to MPAMVPM7_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, MPAMVPM0_EL2.

Virtual PARTID mapping is enabled by MPAMHCR_EL2.EL1_VPMEN for PARTIDs in MPAM1_EL1 and by MPAMHCR_EL2.EL0_VPMEN for PARTIDs in MPAM0_EL1.

A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is valid only when the MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

MPAM

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-103: AARCH64_MPAMVPM1_EL2 bit assignments

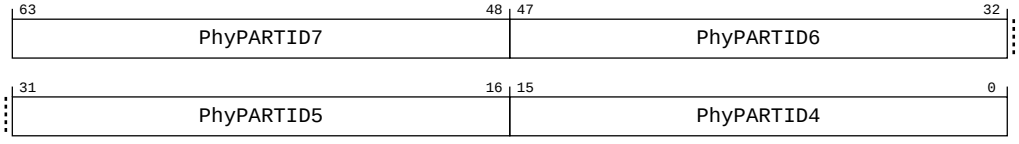


Table A-279: MPAMVPM1_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID7	Virtual PARTID Mapping Entry for virtual PARTID 7. PhyPARTID7 gives the mapping of virtual PARTID 7 to a physical PARTID.	16{x}
[47:32]	PhyPARTID6	Virtual PARTID Mapping Entry for virtual PARTID 6. PhyPARTID6 gives the mapping of virtual PARTID 6 to a physical PARTID.	16{x}
[31:16]	PhyPARTID5	Virtual PARTID Mapping Entry for virtual PARTID 5. PhyPARTID5 gives the mapping of virtual PARTID 5 to a physical PARTID.	16{x}
[15:0]	PhyPARTID4	Virtual PARTID Mapping Entry for virtual PARTID 4. PhyPARTID4 gives the mapping of virtual PARTID 4 to a physical PARTID.	16{x}

Access

MRS <Xt>, MPAMVPM1_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b001

MSR MPAMVPM1_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b001

Accessibility

MRS <Xt>, MPAMVPM1_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'1x1'} then
        X[t, 64] = NVMem[0x948];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if EL3SDDUndef() then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMVPM1_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPM1_EL2;
```

MSR MPAMVPM1_EL2, <Xt>

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

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'1x1'} then
        NVMem[0x948] = X[t, 64];
    elsif EffectiveHCR_EL2_NVx() IN {'xx1'} then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if EL3SDDUndef() then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM1_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMVPM1_EL2 = X[t, 64];
```

A.7 AArch64 Other registers summary

The following summary table provides an overview of all Other registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-282: Other registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SCTLR_EL1	3	0	C1	C0	0	See individual bit resets.	64-bit	System Control Register (EL1)
CPACR_EL1	3	0	C1	C0	2	See individual bit resets.	64-bit	Architectural Feature Access Control Register
ZCR_EL1	3	0	C1	C2	0	See individual bit resets.	64-bit	SVE Control Register (EL1)
SMPRI_EL1	3	0	C1	C2	4	See individual bit resets.	64-bit	Streaming Mode Priority Register
SMCR_EL1	3	0	C1	C2	6	See individual bit resets.	64-bit	SME Control Register (EL1)
SCTLR_EL2	3	4	C1	C0	0	See individual bit resets.	64-bit	System Control Register (EL2)
HCR_EL2	3	4	C1	C1	0	See individual bit resets.	64-bit	Hypervisor Configuration Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
CPTR_EL2	3	4	C1	C1	2	See individual bit resets.	64-bit	Architectural Feature Trap Register (EL2)
HSTR_EL2	3	4	C1	C1	3	See individual bit resets.	64-bit	Hypervisor System Trap Register
HFGRTR_EL2	3	4	C1	C1	4	See individual bit resets.	64-bit	Hypervisor Fine-Grained Read Trap Register
HFGWTR_EL2	3	4	C1	C1	5	See individual bit resets.	64-bit	Hypervisor Fine-Grained Write Trap Register
HFGITR_EL2	3	4	C1	C1	6	See individual bit resets.	64-bit	Hypervisor Fine-Grained Instruction Trap Register
ZCR_EL2	3	4	C1	C2	0	See individual bit resets.	64-bit	SVE Control Register (EL2)
HCRX_EL2	3	4	C1	C2	2	See individual bit resets.	64-bit	Extended Hypervisor Configuration Register
SMPRIMAP_EL2	3	4	C1	C2	5	See individual bit resets.	64-bit	Streaming Mode Priority Mapping Register
SMCR_EL2	3	4	C1	C2	6	See individual bit resets.	64-bit	SME Control Register (EL2)
HDFGRTR_EL2	3	4	C3	C1	4	See individual bit resets.	64-bit	Hypervisor Debug Fine-Grained Read Trap Register
HDFGWTR_EL2	3	4	C3	C1	5	See individual bit resets.	64-bit	Hypervisor Debug Fine-Grained Write Trap Register
HAFGRTR_EL2	3	4	C3	C1	6	See individual bit resets.	64-bit	Hypervisor Activity Monitors Fine-Grained Read Trap Register
SCTLR_EL3	3	6	C1	C0	0	See individual bit resets.	64-bit	System Control Register (EL3)
ZCR_EL3	3	6	C1	C2	0	See individual bit resets.	64-bit	SVE Control Register (EL3)
SMCR_EL3	3	6	C1	C2	6	See individual bit resets.	64-bit	SME Control Register (EL3)

A.8 AArch64 PMU registers summary

The following summary table provides an overview of all PMU registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-283: PMU registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMINTENSET_EL1	3	0	C9	C14	1	See individual bit resets.	64-bit	Performance Monitors Interrupt Enable Set Register
PMINTENCLR_EL1	3	0	C9	C14	2	See individual bit resets.	64-bit	Performance Monitors Interrupt Enable Clear Register
PMMIR_EL1	3	0	C9	C14	6	0x000000000000000A	64-bit	Performance Monitors Machine Identification Register
PMCR_EL0	3	3	C9	C12	0	See individual bit resets.	64-bit	Performance Monitors Control Register
PMCNTENSET_EL0	3	3	C9	C12	1	See individual bit resets.	64-bit	Performance Monitors Count Enable Set Register
PMCNTENCLR_EL0	3	3	C9	C12	2	See individual bit resets.	64-bit	Performance Monitors Count Enable Clear Register
PMOVSCLR_EL0	3	3	C9	C12	3	See individual bit resets.	64-bit	Performance Monitors Overflow Flag Status Clear Register
PMSWINC_EL0	3	3	C9	C12	4	See individual bit resets.	64-bit	Performance Monitors Software Increment Register
PMSELR_EL0	3	3	C9	C12	5	See individual bit resets.	64-bit	Performance Monitors Event Counter Selection Register
PMCEID0_EL0	3	3	C9	C12	6	See individual bit resets.	64-bit	Performance Monitors Common Event Identification Register 0
PMCEID1_EL0	3	3	C9	C12	7	See individual bit resets.	64-bit	Performance Monitors Common Event Identification Register 1
PMCCNTR_EL0	3	3	C9	C13	0	See individual bit resets.	64-bit	Performance Monitors Cycle Count Register
PMXEVTYPER_EL0	3	3	C9	C13	1	See individual bit resets.	64-bit	Performance Monitors Selected Event Type Register
PMXEVCNTR_EL0	3	3	C9	C13	2	See individual bit resets.	64-bit	Performance Monitors Selected Event Count Register
PMUSERENR_EL0	3	3	C9	C14	0	See individual bit resets.	64-bit	Performance Monitors User Enable Register
PMOVSSET_EL0	3	3	C9	C14	3	See individual bit resets.	64-bit	Performance Monitors Overflow Flag Status Set Register
PMEVCNTR0_EL0	3	3	C14	C8	0	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR1_EL0	3	3	C14	C8	1	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR2_EL0	3	3	C14	C8	2	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR3_EL0	3	3	C14	C8	3	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR4_EL0	3	3	C14	C8	4	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR5_EL0	3	3	C14	C8	5	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR6_EL0	3	3	C14	C8	6	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR7_EL0	3	3	C14	C8	7	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR8_EL0	3	3	C14	C9	0	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR9_EL0	3	3	C14	C9	1	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR10_EL0	3	3	C14	C9	2	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR11_EL0	3	3	C14	C9	3	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR12_EL0	3	3	C14	C9	4	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR13_EL0	3	3	C14	C9	5	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR14_EL0	3	3	C14	C9	6	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR15_EL0	3	3	C14	C9	7	See individual bit resets.	64-bit	Performance Monitors Event Count Registers

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMEVCNTR16_EL0	3	3	C14	C10	0	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR17_EL0	3	3	C14	C10	1	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR18_EL0	3	3	C14	C10	2	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR19_EL0	3	3	C14	C10	3	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR20_EL0	3	3	C14	C10	4	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR21_EL0	3	3	C14	C10	5	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR22_EL0	3	3	C14	C10	6	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR23_EL0	3	3	C14	C10	7	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR24_EL0	3	3	C14	C11	0	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR25_EL0	3	3	C14	C11	1	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR26_EL0	3	3	C14	C11	2	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR27_EL0	3	3	C14	C11	3	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR28_EL0	3	3	C14	C11	4	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR29_EL0	3	3	C14	C11	5	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR30_EL0	3	3	C14	C11	6	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVTYPER0_EL0	3	3	C14	C12	0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER1_EL0	3	3	C14	C12	1	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER2_EL0	3	3	C14	C12	2	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER3_EL0	3	3	C14	C12	3	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER4_EL0	3	3	C14	C12	4	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER5_EL0	3	3	C14	C12	5	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER6_EL0	3	3	C14	C12	6	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER7_EL0	3	3	C14	C12	7	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER8_EL0	3	3	C14	C13	0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER9_EL0	3	3	C14	C13	1	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER10_EL0	3	3	C14	C13	2	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER11_EL0	3	3	C14	C13	3	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER12_EL0	3	3	C14	C13	4	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER13_EL0	3	3	C14	C13	5	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER14_EL0	3	3	C14	C13	6	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER15_EL0	3	3	C14	C13	7	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER16_EL0	3	3	C14	C14	0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER17_EL0	3	3	C14	C14	1	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER18_EL0	3	3	C14	C14	2	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER19_EL0	3	3	C14	C14	3	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER20_EL0	3	3	C14	C14	4	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER21_EL0	3	3	C14	C14	5	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER22_EL0	3	3	C14	C14	6	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER23_EL0	3	3	C14	C14	7	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER24_EL0	3	3	C14	C15	0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMEVTYPER25_EL0	3	3	C14	C15	1	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER26_EL0	3	3	C14	C15	2	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER27_EL0	3	3	C14	C15	3	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER28_EL0	3	3	C14	C15	4	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER29_EL0	3	3	C14	C15	5	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER30_EL0	3	3	C14	C15	6	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMCCFILTR_EL0	3	3	C14	C15	7	See individual bit resets.	64-bit	Performance Monitors Cycle Count Filter Register

A.8.1 PMMIR_EL1, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation to software.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

PMU

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1010
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-104: AARCH64_PMMIR_EL1 bit assignments

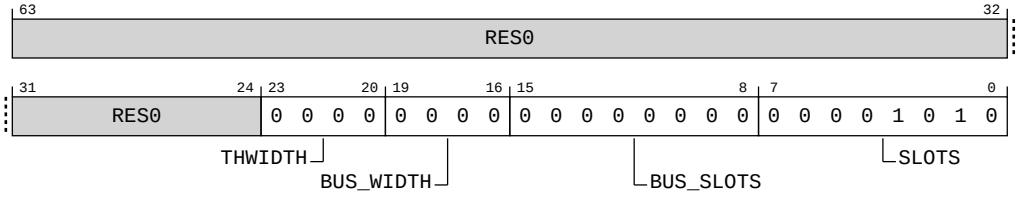


Table A-284: PMMIR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[23:20]	THWIDTH	PMEVTYPER<n>_EL0.TH width. Indicates implementation of the FEAT_PMUv3_TH feature, and, if implemented, the size of the PMEVTYPER<n>_EL0.TH field. **0b0000** FEAT_PMUv3_TH is not implemented.	0b0000
[19:16]	BUS_WIDTH	Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as Log_2(number of bytes), plus one. **0b0000** The information is not available.	0b0000
[15:8]	BUS_SLOTS	Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle. **0x00** The largest value by which the BUS_ACCESS PMU event may increment in one cycle is 0.	0x00
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero. **0x0A** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 10.	0x0A

Access

MRS <Xt>, PMMIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b110

Accessibility

MRS <Xt>, PMMIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMMIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMMIR_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMMIR_EL1;
elsif PSTATE.EL == EL3 then
```

```
    X[t, 64] = PMMIR_EL1;
```

A.8.2 PMCR_EL0, Performance Monitors Control Register

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configurations

AArch64 register PMCR_EL0 bits [31:0] are architecturally mapped to External register B.6.7 PMCR_EL0, Performance Monitors Control Register on page 773 bits [31:0].

Attributes

Width

64

Functional group

PMU

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 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 A-105: AARCH64_PMCR_EL0 bit assignments

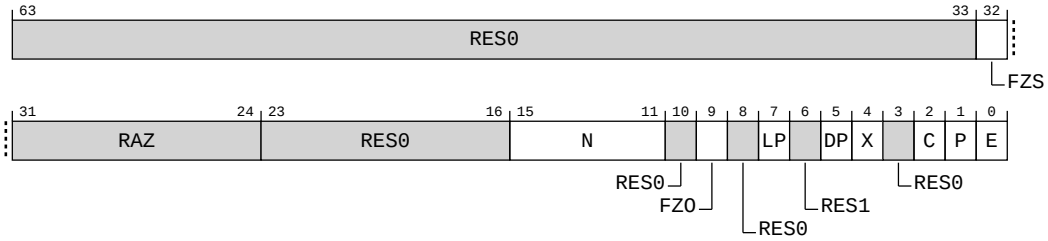


Table A-286: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:33]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[32]	FZS	Freeze-on-SPE event. Stop counters when PMBLIMITR_EL1.{PMFZ,E} is {1,1} and PMBSR_EL1.S is 1. **0b0** Do not freeze on a Statistical Profiling Buffer Management event. **0b1** Affected counters do not count following a Statistical Profiling Buffer Management event.	x
[31:24]	**RAZ**	Reserved	**RAZ**
[23:16]	**RES0**	Reserved	**RES0**
[15:11]	N	Indicates the number of event counters implemented. This value is in the range of `0b00000`-`0b11111`. If the value is `0b00000`, then only PMCCNTR_EL0 is implemented. If the value is `0b11111`, then PMCCNTR_EL0 and 31 event counters are implemented. When EL2 is implemented and enabled for the current Security state, reads of this field from EL1 and EL0 return the Effective value of MDCR_EL2.HPMN. **0b00110** Six event counters and the cycle counter implemented. **0b11111** Thirty-one event counters and the cycle counter implemented.	The reset values can be the following: `0b00110`, `0b11111`, respective to the value.
[10]	**RES0**	Reserved	**RES0**
[9]	FZO	Freeze-on-overflow. Stop event counters on overflow. **0b0** Do not freeze on overflow. **0b1** Affected counters do not count when PMOVSCLR_EL0[m] is 1 for any event counter PMEVCNTR<m>_EL0 in the first range.	x
[8]	**RES0**	Reserved	**RES0**
[7]	LP	Long event counter enable. Determines when unsigned overflow is recorded by PMOVSCLR_EL0.P[n]. **0b0** Event counter overflow on increment that causes unsigned overflow of PMEVCNTR<n>_EL0[31:0]. **0b1** Event counter overflow on increment that causes unsigned overflow of PMEVCNTR<n>_EL0[63:0].	x
[6]	**RES1**	Reserved	**RES1**

Bits	Name	Description	Reset
[5]	DP	Disable cycle counter when event counting is prohibited. **0b0** Cycle counting by PMCCNTR_EL0 is not affected by this mechanism. **0b1** Cycle counting by PMCCNTR_EL0 is disabled in prohibited regions and when event counting is frozen: • If FEAT_PMUv3p1 is implemented, EL2 is implemented, and MDCR_EL2.HPMD is 1, then cycle counting by PMCCNTR_EL0 is disabled at EL2. • If FEAT_SPE_DPFZS is implemented and event counting is frozen by PMCR_EL0.FZS, then cycle counting by PMCCNTR_EL0 is disabled. • If FEAT_PMUv3p7 is implemented and event counting is frozen by PMCR_EL0.FZO, then cycle counting by PMCCNTR_EL0 is disabled. • If FEAT_PMUv3p7 is implemented, EL3 is implemented, and MDCR_EL3.MPMX is 1, then cycle counting by PMCCNTR_EL0 is disabled at EL3. • If EL3 is implemented, MDCR_EL3.SPME is 0, and either FEAT_PMUv3p7 is not implemented or MDCR_EL3.MPMX is 0, then cycle counting by PMCCNTR_EL0 is disabled at EL3 and in Secure state.	x
[4]	X	**When ImpDefBool("the implementation includes a PMU event export bus** Enable export of events in an **IMPLEMENTATION DEFINED** PMU event export bus. **0b0** Do not export events. **0b1** Export events where not prohibited. **Otherwise** RAZ/WI	xxxx
[3]	**RES0**	Reserved	**RES0**
[2]	C	Cycle counter reset. The effects of writing to this field are: **0b0** No action. **0b1** Reset PMCCNTR_EL0 to zero. Access to this field is: WO/**RAZ**	0b0
[1]	P	Event counter reset. **0b0** No action. **0b1** Reset all affected event counters PMEVCNTR<n>_EL0 to zero. Access to this field is: WO/**RAZ**	0b0

Bits	Name	Description	Reset
[0]	E	Enable. **0b0** Affected counters are disabled and do not count. **0b1** Affected counters are enabled by PMCNTENSET_EL0.	0b0

Access

MRS <Xt>, PMCR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

MSR PMCR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

Accessibility

MRS <Xt>, PMCR_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCR_EL0;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCR_EL0;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif MDCR_EL3.TPM == '1' then
```

```
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMCR_EL0;
```

MSR PMCR_EL0, <Xt>

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMCR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMCR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMCR_EL0 = X[t, 64];
```

A.8.3 PMCEID0_EL0, Performance Monitors Common Event Identification Register 0

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the ranges `0x0000` to `0x001F` and `0x4000` to `0x401F`.

For more information about the Common events and the use of the PMCEID<n>_EL0 registers see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

AArch64 register PMCEID0_EL0 bits [31:0] are architecturally mapped to External register B.6.9 PMCEID0, Performance Monitors Common Event Identification register 0 on page 779 bits [31:0].

AArch64 register PMCEID0_EL0 bits [63:32] are architecturally mapped to External register B.6.11 PMCEID2, Performance Monitors Common Event Identification register 2 on page 789 bits [31:0].

Attributes

Width

64

Functional group

PMU

Access type

RO

Reset value

```
xxxx 1111 xxxx 1111 0001 101x x111 1111 0111 1111 1111 1111 0110 1111 0011 1111
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-106: AARCH64_PMCEID0_EL0 bit assignments

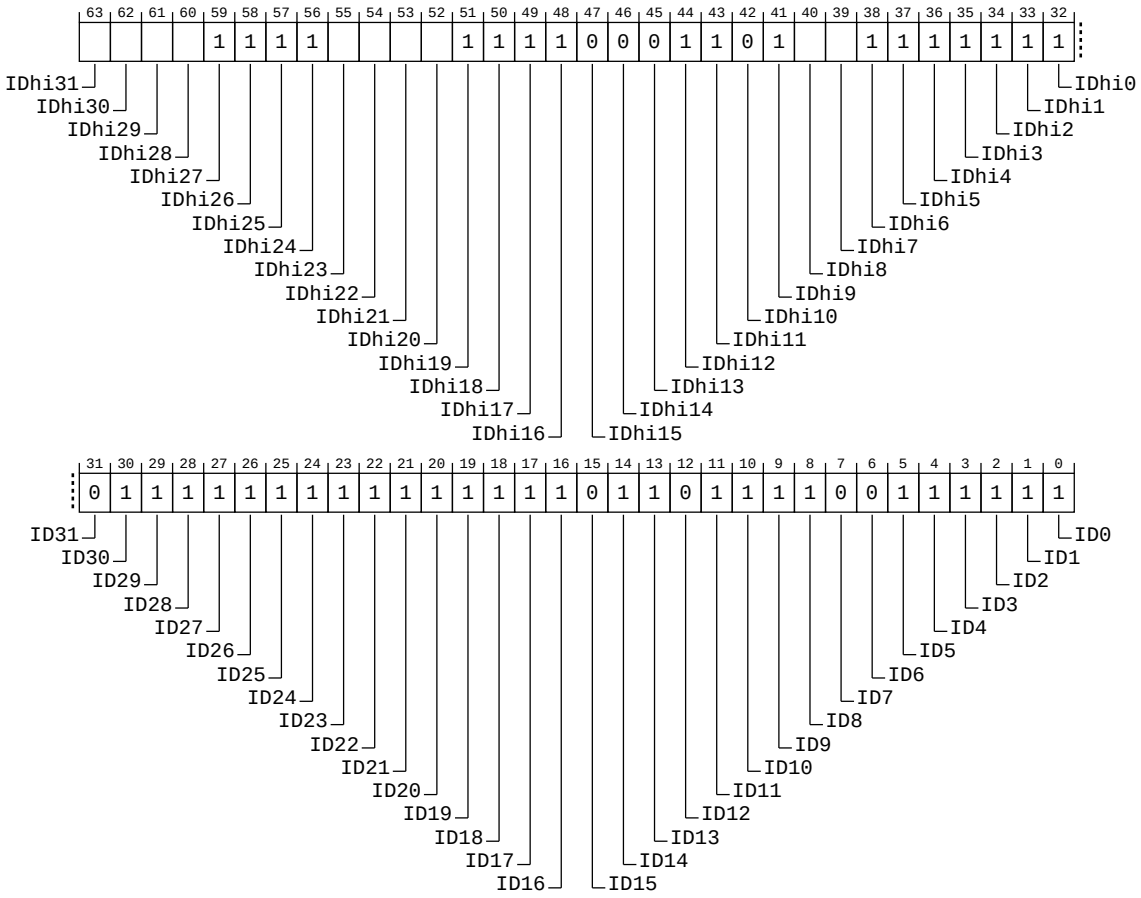


Table A-289: PMCEID0_EL0 bit descriptions

Bits	Name	Description	Reset
[63]	IDhi31	IDhi[n] corresponds to Common event (`0x4000` + n). For each bit: **`0b0`** The Common event is not implemented, or not counted. **`0b1`** The Common event is implemented.	The reset values can be the following: `0b0`, `0b1`, respective to the value.

Bits	Name	Description	Reset
[62]	IDhi30	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[61]	IDhi29	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[60]	IDhi28	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[59]	IDhi27	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[58]	IDhi26	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[57]	IDhi25	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[56]	IDhi24	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[55]	IDhi23	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[54]	IDhi22	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[53]	IDhi21	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[52]	IDhi20	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[51]	IDhi19	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[50]	IDhi18	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[49]	IDhi17	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[48]	IDhi16	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[47]	IDhi15	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[46]	IDhi14	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[45]	IDhi13	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[44]	IDhi12	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[43]	IDhi11	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[42]	IDhi10	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[41]	IDhi9	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[40]	IDhi8	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[39]	IDhi7	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[38]	IDhi6	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[37]	IDhi5	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[36]	IDhi4	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[35]	IDhi3	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[34]	IDhi2	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[33]	IDhi1	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[32]	IDhi0	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[31]	ID31	ID[n] corresponds to Common event n. For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[30]	ID30	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[29]	ID29	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[28]	ID28	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[27]	ID27	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[26]	ID26	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[25]	ID25	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[24]	ID24	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[23]	ID23	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[22]	ID22	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[21]	ID21	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[20]	ID20	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[19]	ID19	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[18]	ID18	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[17]	ID17	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[16]	ID16	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[15]	ID15	ID[n] corresponds to Common event n. For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[14]	ID14	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[13]	ID13	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[12]	ID12	ID[n] corresponds to Common event n. For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[11]	ID11	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[10]	ID10	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[9]	ID9	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[8]	ID8	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[7]	ID7	ID[n] corresponds to Common event n. For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[6]	ID6	ID[n] corresponds to Common event n. For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[5]	ID5	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[4]	ID4	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[3]	ID3	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[2]	ID2	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[1]	ID1	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[0]	ID0	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1

Access

`MRS <Xt>, PMCEID0_EL0`

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b110

Accessibility

MRS <Xt>, PMCEID0_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID0_EL0;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMCEIDn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID0_EL0;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID0_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMCEID0_EL0;
```

A.8.4 PMCEID1_EL0, Performance Monitors Common Event Identification Register 1

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the ranges `0x0020` to `0x003F` and `0x4020` to `0x403F`.

For more information about the Common events and the use of the PMCEID<n>_EL0 registers see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

AArch64 register PMCEID1_EL0 bits [31:0] are architecturally mapped to External register B.6.10 PMCEID1, Performance Monitors Common Event Identification register 1 on page 784 bits [31:0].

AArch64 register PMCEID1_EL0 bits [63:32] are architecturally mapped to External register B.6.12 PMCEID3, Performance Monitors Common Event Identification register 3 on page 796 bits [31:0].

Attributes

Width

64

Functional group

PMU

Access type

RO

Reset value

```
0000 0000 xx00 x000 xxxx xxxx x111 x111 1111 1110 1111 0010 1010 1110 0111 1111
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
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-107: AARCH64_PMCEID1_EL0 bit assignments

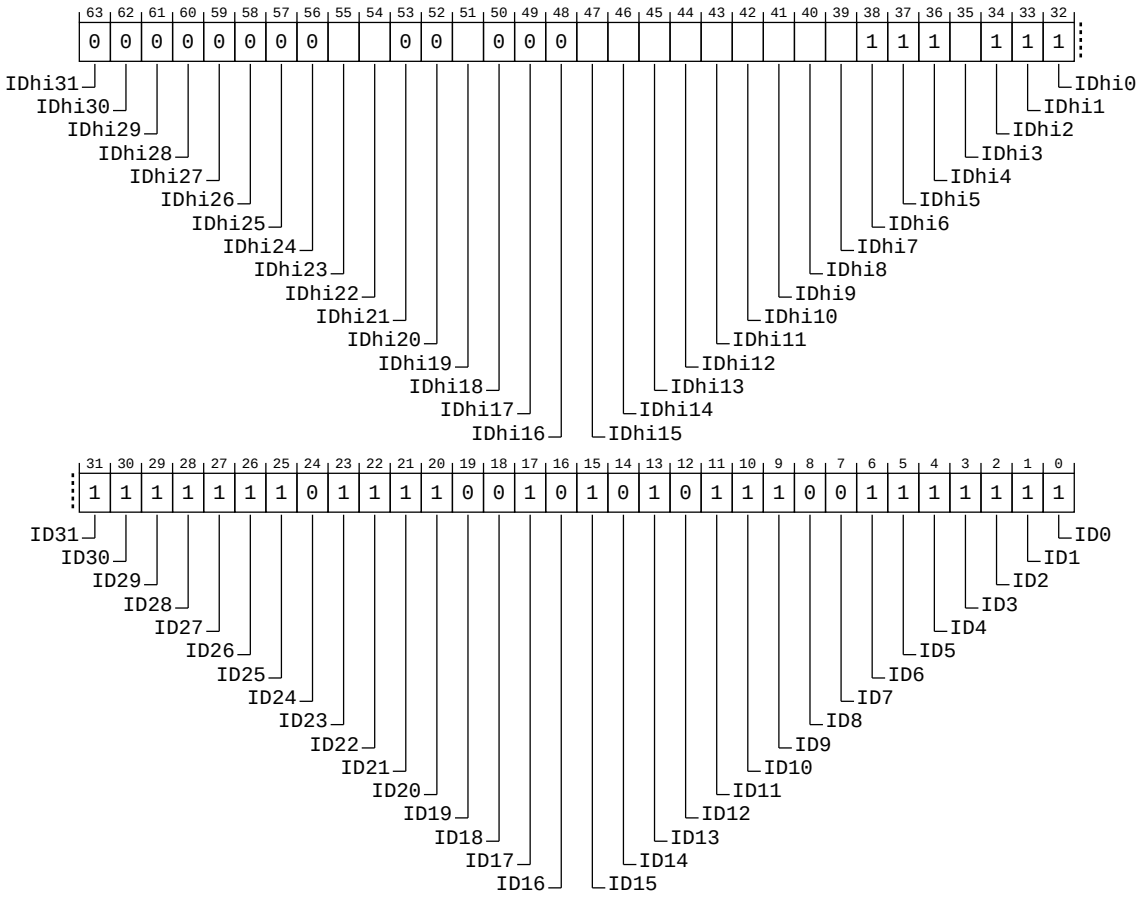


Table A-291: PMCEID1_EL0 bit descriptions

Bits	Name	Description	Reset
[63]	IDhi31	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[62]	IDhi30	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[61]	IDhi29	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[60]	IDhi28	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[59]	IDhi27	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[58]	IDhi26	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[57]	IDhi25	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[56]	IDhi24	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[55]	IDhi23	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.

Bits	Name	Description	Reset
[54]	IDhi22	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[53]	IDhi21	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[52]	IDhi20	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[51]	IDhi19	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[50]	IDhi18	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[49]	IDhi17	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[48]	IDhi16	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[47]	IDhi15	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[46]	IDhi14	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[45]	IDhi13	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[44]	IDhi12	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[43]	IDhi11	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.

Bits	Name	Description	Reset
[42]	IDhi10	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[41]	IDhi9	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[40]	IDhi8	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[39]	IDhi7	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[38]	IDhi6	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[37]	IDhi5	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[36]	IDhi4	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[35]	IDhi3	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[34]	IDhi2	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[33]	IDhi1	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[32]	IDhi0	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[31]	ID31	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[30]	ID30	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[29]	ID29	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[28]	ID28	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[27]	ID27	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[26]	ID26	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[25]	ID25	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[24]	ID24	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[23]	ID23	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[22]	ID22	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[21]	ID21	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[20]	ID20	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[19]	ID19	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[18]	ID18	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[17]	ID17	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[16]	ID16	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[15]	ID15	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[14]	ID14	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[13]	ID13	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[12]	ID12	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[11]	ID11	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[10]	ID10	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[9]	ID9	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[8]	ID8	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[7]	ID7	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[6]	ID6	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[5]	ID5	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[4]	ID4	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[3]	ID3	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[2]	ID2	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[1]	ID1	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[0]	ID0	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1

Access

MRS <Xt>, PMCEID1_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b111

Accessibility

MRS <Xt>, PMCEID1_EL0

```
if PSTATE.EL == EL0 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID1_EL0;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
```

```
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMCEIDn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID1_EL0;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif MDCR_EL3.TPM == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID1_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMCEID1_EL0;
```

A.9 AArch64 PPM registers summary

The following summary table provides an overview of all PPM 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-293: PPM registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CPUPPMPDPCR_EL1	3	0	C15	C2	4	See individual bit resets.	64-bit	Performance and Power Management PDP Control Register
IMP_CPUPPMCR_EL3	3	6	C15	C2	0	See individual bit resets.	64-bit	Global Performance and Power Management Configuration Register
IMP_CPUMPMMCR_EL3	3	6	C15	C2	1	See individual bit resets.	64-bit	Global MPMM Control Register
IMP_CPUACTMCTL_EL3	3	6	C15	C2	2	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR4_EL3	3	6	C15	C2	4	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR5_EL3	3	6	C15	C2	5	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR6_EL3	3	6	C15	C2	6	See individual bit resets.	64-bit	CPU Power Performance Management Control Register

A.9.1 IMP_CPUPPMPDPCR_EL1, Performance and Power Management PDP Control Register

Provides **IMPLEMENTATION DEFINED** control of the Performance Defined Power (PDP) feature

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

PPM registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-108: AARCH64_IMP_CPUPPMPDPCR_EL1 bit assignments

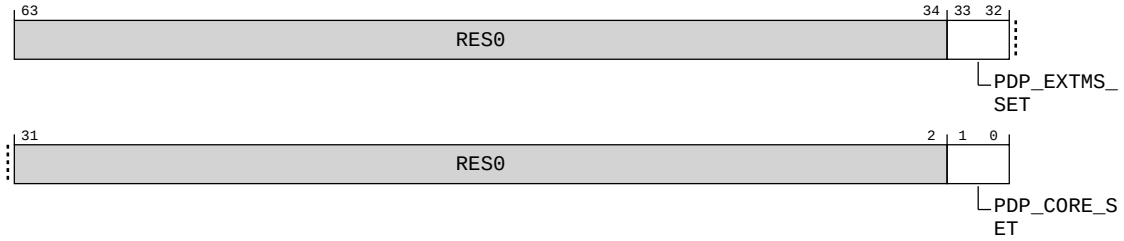


Table A-294: IMP_CPUPPMPDPCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:34]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[33:32]	PDP_EXTMS_SET	External memory system PDP Aggressiveness **0b00** Disable PDP **0b01** Enable PDP at low aggressiveness **0b10** Enable PDP at medium aggressiveness **0b11** Enable PDP at high aggressiveness	0b00
[31:2]	**RES0**	Reserved	**RES0**
[1:0]	PDP_CORE_SET	Core PDP Aggressiveness **0b00** Disable PDP **0b01** Enable PDP at low aggressiveness **0b10** Enable PDP at medium aggressiveness **0b11** Enable PDP at high aggressiveness	0b00

Access

MRS <Xt>, S3_0_C15_C2_4

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b100

MSR S3_0_C15_C2_4, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b100

Accessibility

MRS <Xt>, S3_0_C15_C2_4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUPPMPDPCR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPPMPDPCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPPMPDPCR_EL1;
```

```
MSR S3_0_C15_C2_4, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.PDPEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.PDPEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPPMPDPCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.PDPEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPPMPDPCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMPDPCR_EL1 = X[t, 64];
```

A.9.2 IMP_CPUPPMCR_EL3, Global Performance and Power Management Configuration Register

Provides **IMPLEMENTATION DEFINED** control and discovery of the Performance and Power Management (PPM) features

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

PPM registers

Access type

RW

Reset value

```
1xxx xxxx xxxx xxxx xxx0 xxxx xx00 0000 xxxx xxxx xxxx x111 xxxx x011 xxxx xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-109: AARCH64_IMP_CPUPPMCR_EL3 bit assignments

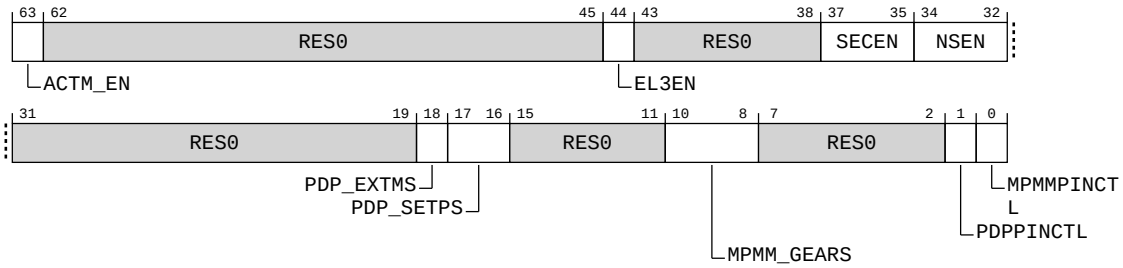


Table A-297: IMP_CPUPPMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63]	ACTM_EN	Activity Meter Enable **0b1** The Activity Meter is enabled. This field is read only.	0b1
[62:45]	**RES0**	Reserved	**RES0**
[44]	EL3EN	EL3 Activity Meter Count Enable. Enables Activity Meter update in EL3 - Root or EL3 CPU Activity is counted if this bit is set **0b0** Inhibit update of Activity Meter when core is in EL3 **0b1** Allow update of Activity Meter when core is in EL3	0b0
[43:38]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[37:35]	SECEN	Secure ELx Activity Meter Count Enable. Enables Activity Meter update in Secure ELx. - Each bit if set will enable counting in that particular secure EL **0b000** Inhibit update of Activity Meter when core is in Secure EL0/EL1/EL2 **0b001** Inhibit update of Activity Meter when core is in Secure EL1/EL2, and allow update in Secure EL0 **0b010** Inhibit update of Activity Meter when core is in Secure EL0/EL2, and allow update in Secure EL1 **0b011** Inhibit update of Activity Meter when core is in Secure EL2, and allow update in Secure EL0/EL1 **0b100** Inhibit update of Activity Meter when core is in Secure EL0/EL1, and allow update in Secure EL2 **0b101** Inhibit update of Activity Meter when core is in Secure EL1, and allow update in Secure EL0/EL2 **0b110** Inhibit update of Activity Meter when core is in Secure EL0, and allow update in Secure EL1/EL2 **0b111** Allow update of Activity Meter when core is in Secure EL0/EL1/EL2	0b000

Bits	Name	Description	Reset
[34:32]	NSEN	Non-secure ELx Activity Meter Count Enable. Enables Activity Meter update in Non-secure ELx. - Each bit if set will enable counting in that particular non-secure EL **0b000** Inhibit update of Activity Meter when core is in Non-secure EL0/EL1/EL2 **0b001** Inhibit update of Activity Meter when core is in Non-secure EL1/EL2, and allow update in Non-secure EL0 **0b010** Inhibit update of Activity Meter when core is in Non-secure EL0/EL2, and allow update in Non-secure EL1 **0b011** Inhibit update of Activity Meter when core is in Non-secure EL2, and allow update in Non-secure EL0/EL1 **0b100** Inhibit update of Activity Meter when core is in Non-secure EL0/EL1, and allow update in Non-secure EL2 **0b101** Inhibit update of Activity Meter when core is in Non-secure EL1, and allow update in Non-secure EL0/EL2 **0b110** Inhibit update of Activity Meter when core is in Non-secure EL0, and allow update in Non-secure EL1/EL2 **0b111** Allow update of Activity Meter when core is in Non-secure EL0/EL1/EL2	0b000
[31:19]	**RES0**	Reserved	**RES0**
[18]	PDP_EXTMS	External memory system PDP control **0b0** Independent external memory system PDP control is not implemented **0b1** Independent external memory system PDP control is implemented	0b1
[17:16]	PDP_SETPS	Number of PDP Setpoints Implemented **0b11** 3 PDP setpoints are implemented	0b11
[15:11]	**RES0**	Reserved	**RES0**
[10:8]	MPMM_GEARS	Number of MPMM Gears Implemented **0b011** 3 MPMM gears are implemented	0b011
[7:2]	**RES0**	Reserved	**RES0**
[1]	PDPPINCTL	PDP Pin Control Enabled **0b0** PDP control through SPR and utility bus **0b1** PDP control through pin only	0b0

Bits	Name	Description	Reset
[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_C2_0

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b000

MSR S3_6_C15_C2_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b000

Accessibility

MRS <Xt>, S3_6_C15_C2_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPPMCR_EL3;
```

MSR S3_6_C15_C2_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR_EL3 = X[t, 64];
```

A.9.3 IMP_CPUMPMMCR_EL3, Global MPMM Control Register

Provides **IMPLEMENTATION DEFINED** control of the Maximum Power Mitigation Mechanism (MPMM) feature

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

PPM registers

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 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 A-110: AARCH64_IMP_CPUMPMMCR_EL3 bit assignments

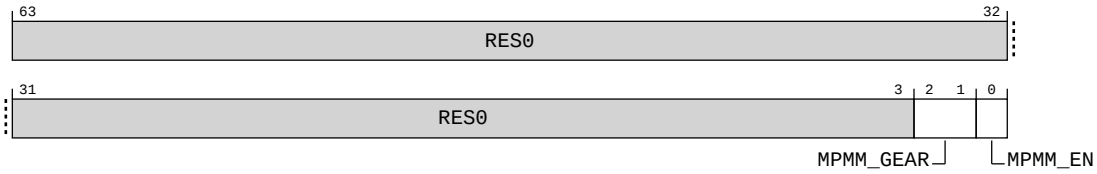


Table A-300: IMP_CPUMPMMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:3]	**RES0**	Reserved	**RES0**
[2:1]	MPMM_GEAR	MPMM Gear Select **0b00** Select MPMM Gear 0 **0b01** Select MPMM Gear 1 **0b10** Select MPMM Gear 2 **0b11** Select MPMM Gear 3	0b00

Bits	Name	Description	Reset
[0]	MPMM_EN	MPMM Global Enable **0b0** MPMM is disabled **0b1** MPMM is enabled	0b0

Access

MRS <Xt>, S3_6_C15_C2_1

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b001

MSR S3_6_C15_C2_1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b001

Accessibility

MRS <Xt>, S3_6_C15_C2_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUMPMMCR_EL3;
```

MSR S3_6_C15_C2_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUMPMMCR_EL3 = X[t, 64];
```

A.9.4 IMP_CPUACTMCTL_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

PPM 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-111: AARCH64_IMP_CPUACTMCTL_EL3 bit assignments

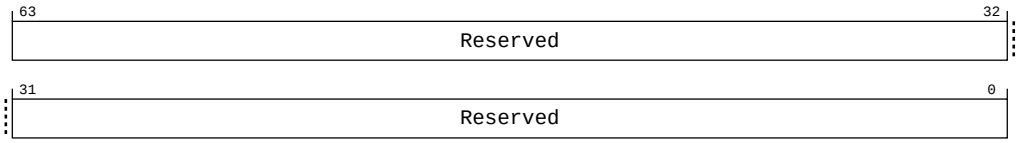


Table A-303: IMP_CPUACTMCTL_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_2

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b010

MSR S3_6_C15_C2_2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b010

Accessibility

`MRS <Xt>, S3_6_C15_C2_2`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTMCTL_EL3;
```

`MSR S3_6_C15_C2_2, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUACTMCTL_EL3 = X[t, 64];
```

A.9.5 IMP_CPUPPMCR4_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

PPM 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-112: AARCH64_IMP_CPUPPMCR4_EL3 bit assignments

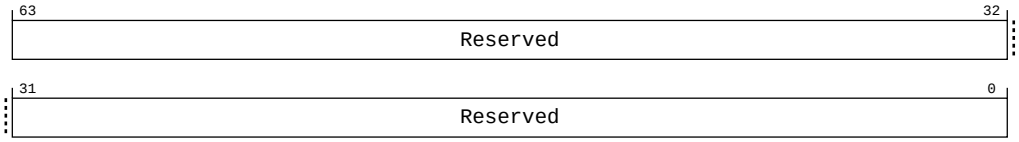


Table A-306: IMP_CPUPPMCR4_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_4

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b100

MSR S3_6_C15_C2_4, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b100

Accessibility

MRS <Xt>, S3_6_C15_C2_4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPPMCR4_EL3;
```

MSR S3_6_C15_C2_4, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR4_EL3 = X[t, 64];
```

A.9.6 IMP_CPUPPMCR5_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

PPM 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-113: AARCH64_IMP_CPUPPMCR5_EL3 bit assignments

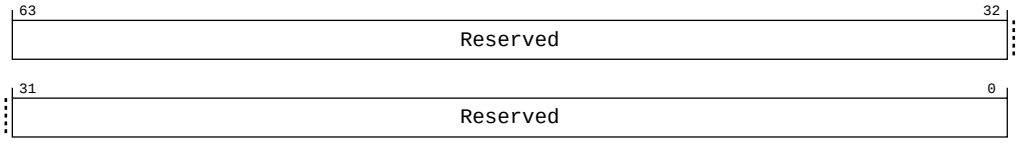


Table A-309: IMP_CPUPPMCR5_EL3 bit descriptions

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

Access

```
MRS <Xt>, S3_6_C15_C2_5
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b101

```
MSR S3_6_C15_C2_5, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b101

Accessibility

```
MRS <Xt>, S3_6_C15_C2_5
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPPMCR5_EL3;
```

```
MSR S3_6_C15_C2_5, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR5_EL3 = X[t, 64];
```

A.9.7 IMP_CPUPPMCR6_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

PPM 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-114: AARCH64_IMP_CPUPPMCR6_EL3 bit assignments

Table A-312: IMP_CPUPPMCR6_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_6

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b110

MSR S3_6_C15_C2_6, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b110

Accessibility

MRS <Xt>, S3_6_C15_C2_6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPPMCR6_EL3;
```

MSR S3_6_C15_C2_6, <Xt>

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

```
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR6_EL3 = X[t, 64];
```

A.10 AArch64 RAS registers summary

The following summary table provides an overview of all RAS registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-315: RAS registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ERRIDR_EL1	3	0	C5	C3	0	0x0000000000000002	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	0x005100008010A9A2	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	0x0000000040000062	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
DISR_EL1	3	0	C12	C1	1	See individual bit resets.	64-bit	Deferred Interrupt Status Register
VSESR_EL2	3	4	C5	C2	3	See individual bit resets.	64-bit	Virtual SError Exception Syndrome Register
VDISR_EL2	3	4	C12	C1	1	See individual bit resets.	64-bit	Virtual Deferred Interrupt Status Register (EL2)

A.10.1 ERRIDR_EL1, Error Record ID Register

Defines the highest numbered index of the error records that can be accessed through the Error Record System registers.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Access type

RO

Reset value

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

Bit descriptions

Figure A-115: AARCH64_ERRIDR_EL1 bit assignments

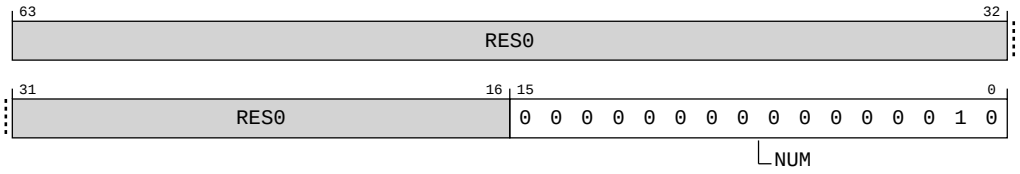


Table A-316: ERRIDR_EL1 bit descriptions

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. **0x0002** Two records present.	0x0002

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;
```

A.10.2 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

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 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-116: AARCH64_ERRSELR_EL1 bit assignments

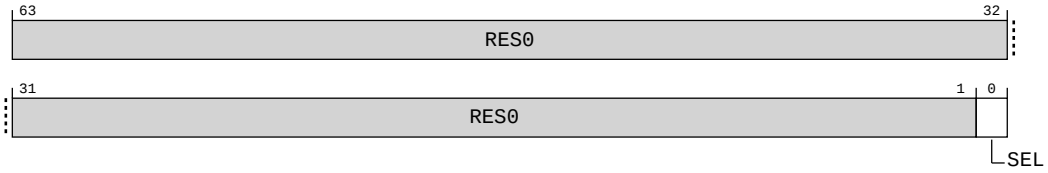


Table A-318: ERRSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	**RES0**	Reserved	**RES0**
[0]	SEL	**0b0** Selects record 0, containing errors from DSU RAMs **0b1** Selects record 1, containing errors from Core RAMs	0b0

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
    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.10.3 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

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 A-117: AARCH64_ERXFR_EL1 bit assignments

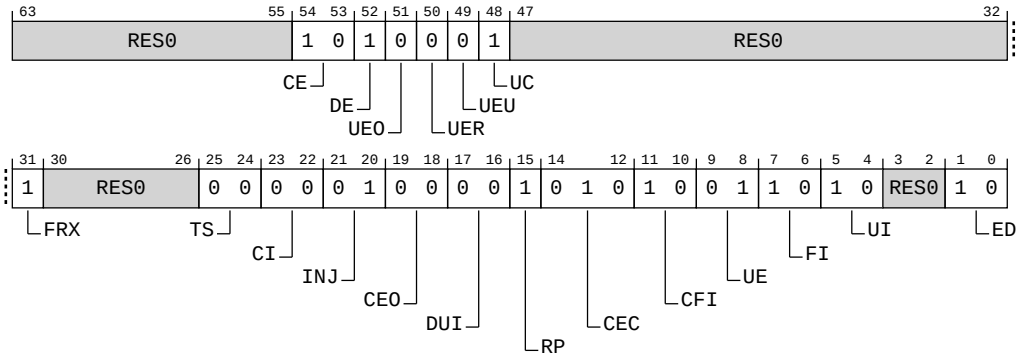


Table A-321: ERXFR_EL1 bit descriptions

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 ERXSTATUS_EL1.CE to 0b10.	0b10
[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	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. **0b0** Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors.	0b0
[48]	UC	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. **0b1** Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors.	0b1
[47:32]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[31]	FRX	Feature Register extension. Defines whether ERXFR_EL1[63:48] are architecturally defined. **0b1** ERXFR_EL1[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, ERXMISC3_EL1 is used as the timestamp register, and, if it is, the timebase used by the timestamp. **0b00** The node does not support a timestamp register.	0b00
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented. **0b00** Does not support the critical error interrupt. ERXCTLR_EL1.CI is **RES0**.	0b00
[21:20]	INJ	Fault Injection Extension. Indicates whether the RAS Common Fault Injection Model Extension is implemented. **0b01** The node implements the RAS Common Fault Injection Model Extension. See ERXPFGF_EL1 for more information.	0b01
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior when a second Corrected error is detected after a first Corrected error has been recorded by an error record <m> owned by the node. **0b00** Counts Corrected errors if a counter is implemented. Keeps the previous error syndrome. If the counter overflows, or no counter is implemented, then ERXSTATUS_EL1.OF is set to 0b1.	0b00
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the control for enabling error recovery interrupts on deferred errors are implemented. **0b00** Does not support the control for enabling error recovery interrupts on deferred errors. ERXCTLR_EL1.DUI is **RES0**.	0b00
[15]	RP	Repeat counter. Indicates whether the node implements the repeat Corrected error counter in ERXMISC0_EL1 for each error record <m> owned by the node that implements the standard Corrected error counter. **0b1** A first (repeat) counter and a second (other) counter are implemented. The repeat counter is the same size as the primary error counter.	0b1
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter (CE counter) mechanisms in ERXMISC0_EL1 for each error record <m> owned by the node that can record countable errors. **0b010** Implements an 8-bit Corrected error counter in ERXMISC0_EL1[39:32].	0b010
[11:10]	CFI	Fault handling interrupt for corrected errors. Indicates whether the control for enabling fault handling interrupts on corrected errors are implemented. **0b10** Control for enabling fault handling interrupts on corrected errors is supported and controllable using ERXCTLR_EL1.CFI.	0b10
[9:8]	UE	In-band uncorrected error reporting. Indicates whether the in-band uncorrected error reporting (External Aborts) and associated controls are implemented. **0b01** In-band uncorrected error reporting (External Aborts) is supported and always enabled. ERXCTLR_EL1.UE is **RES0**.	0b01

Bits	Name	Description	Reset
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented. **0b10** Fault handling interrupt is supported and controllable using ERXCTLR_EL1.FI.	0b10
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented. **0b10** Error handling interrupt is supported and controllable using ERXCTLR_EL1.UI.	0b10
[3:2]	**RES0**	Reserved	**RES0**
[1:0]	ED	Error reporting and logging. Indicates whether error record <n> is the first record owned the node, and, if so, whether it implements the controls for enabling and disabling error reporting and logging. **0b10** Error reporting and logging is controllable using ERXCTLR_EL1.ED.	0b10

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
        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.10.4 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

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 A-118: AARCH64_ERXCTLR_EL1 bit assignments

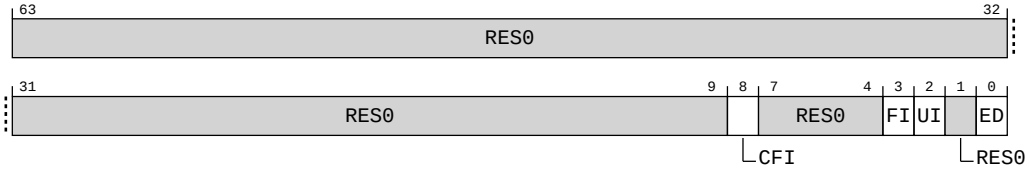


Table A-323: ERXCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:9]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[8]	CFI	Fault handling interrupt for Corrected errors enable. This control applies to errors arising from both reads and writes. The fault handling interrupt is generated when one of the standard CE counters on ERXMISC0_EL1 overflows and the overflow bit is set. The possible values are: **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. Cold reset only. Unaffected by Warm reset	0b0
[7:4]	**RES0**	Reserved	**RES0**
[3]	FI	Fault handling interrupt enable. This control applies to errors arising from both reads and writes. The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. The possible values are: **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. Cold reset only. Unaffected by Warm reset	0b0
[2]	UI	Uncorrected error recovery interrupt enable. This control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred. **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. Cold reset only. Unaffected by Warm reset	0b0
[1]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[0]	ED	Error Detection and correction enable. The possible values are: **0b0** Error detection and correction disabled. **0b1** Error detection and correction enabled. Cold reset only. Unaffected by Warm reset	0b0

Access

`MRS <Xt>, ERXCTLR_EL1`

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;
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.10.5 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

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0000 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.

Bit descriptions

Figure A-119: AARCH64_ERXSTATUS_EL1 bit assignments

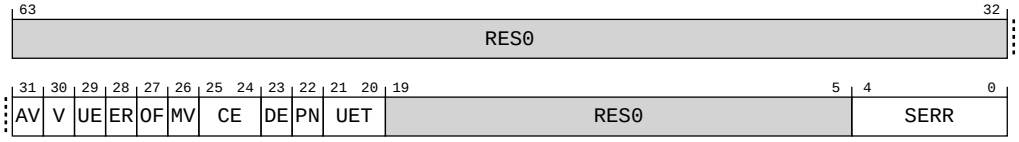


Table A-326: ERXSTATUS_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	AV	Address Valid. The possible values are: **0b0** ERXADDR_EL1 not valid. **0b1** ERXADDR_EL1 contains an address associated with the highest priority error recorded by this record. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0
[30]	V	Status Register Valid. The possible values are: **0b0** ERXSTATUS_EL1 not valid. **0b1** ERXSTATUS_EL1 valid. At least one error has been recorded. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0

Bits	Name	Description	Reset
[29]	UE	Uncorrected Error. The possible values are: **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 ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0
[28]	ER	Error Reported. The possible values are: **0b0** No in-band error (External Abort) reported. **0b1** An External Abort was signaled by the node to the requester making the access or other transaction. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset **Note:** An External Abort signaled by the node might be masked and not generate any exception.	0b0
[27]	OF	Overflow. The possible values are: **0b0** If UE == 1, then no error status for an Uncorrected error has been discarded. If UE == 0 and DE == 1, then no error status for a Deferred error has been discarded. If UE == 0, DE == 0, and CE !== 0b00, then the corrected error counter has not overflowed. **0b1** More than one error has occurred and so details of the other error have been discarded. When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. Cold reset only. Unaffected by Warm reset This bit is read/write-one-to-clear.	0b0

Bits	Name	Description	Reset
[26]	MV	Miscellaneous Registers Valid. The possible values are: **0b0** ERXMISC<m>_EL1 not valid. **0b1** This bit indicates that the ERXMISC<m>_EL1 registers contain additional information for an error recorded by this record. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset **Note:** If the ERXMISC<m>_EL1 registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded.	0b0
[25:24]	CE	Corrected Error. The possible values are: **0b00** No errors were corrected. **0b01** At least one transient error was corrected. **0b10** At least one error was corrected. **0b11** At least one persistent error was corrected. When clearing ERXSTATUS_EL1.V to 0b0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. This field is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. This field is read/write-ones-to-clear. Writing a value other than all-zeros or all-ones sets this field to an **UNKNOWN** value. Cold reset only. Unaffected by Warm reset	0b00
[23]	DE	Deferred Error. The possible values are: **0b0** No errors were deferred. **0b1** At least one error was not corrected and deferred. When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0

Bits	Name	Description	Reset
[22]	PN	Poison. The value is: **0b0** This core cannot distinguish a poisoned value from a corrupted value. When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if any of the following are true: • ERXSTATUS_EL1.V == 0b0. • ERXSTATUS_EL1.{DE,UE} == {0,0}. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0
[21:20]	UET	Uncorrected Error Type. The value is: **0b00** Uncorrected error, Uncontainable error (UC). Cold reset only. Unaffected by Warm reset	0b00
[19:5]	**RES0**	Reserved	**RES0**
[4:0]	SERR	Primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry. The possible values are: **0b00000** No error **0b00010** ECC error from internal data buffer. **0b00110** ECC error on cache data RAM. **0b00111** ECC error on cache tag or dirty RAM. **0b01000** Parity error on TLB data RAM. **0b10010** Error response for a cache copyback. **0b10101** Deferred error from completer not supported at the consumer. For example, poisoned data received from a completer by a requester that cannot defer the error further. Cold reset only. Unaffected by Warm reset	0b00000

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
    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];
```

A.10.6 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

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-120: AARCH64_ERXADDR_EL1 bit assignments

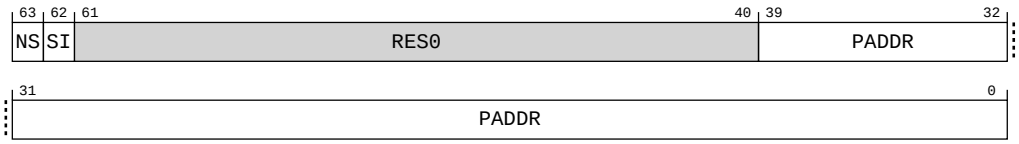


Table A-329: ERXADDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure attribute. **0b0** ERR<n>ADDR.PADDR is a Secure address. **0b1** ERR<n>ADDR.PADDR is a Non-secure address.	x
[62]	SI	Secure Incorrect. Indicates whether ERR<n>ADDR.NS is valid. **0b0** ERR<n>ADDR.NS is correct. That is, it matches the programmers' view of the Non-secure attribute for the recorded location. **0b1** ERR<n>ADDR.NS might not be correct, and might not match the programmers' view of the Non-secure attribute for the recorded location.	x
[61:40]	**RES0**	Reserved	**RES0**
[39:0]	PADDR	Physical Address [39:0]. Address of the recorded location	40{x}

Access

```
MRS <Xt>, ERXADDR_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b011

```
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);
    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.10.7 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

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0100 0000 0000 0000 0000 0000 0110 0010
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure A-121: AARCH64_ERXPFGF_EL1 bit assignments

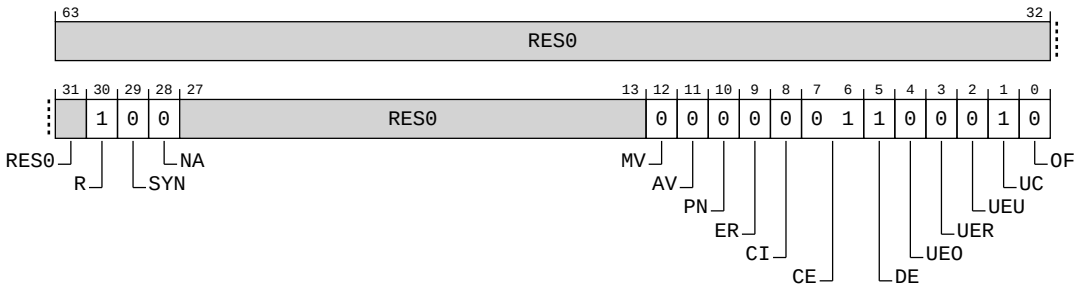


Table A-332: ERXPFGF_EL1 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30]	R	Restartable bit. When it reaches zero, the Error Generation Counter restarts from the ERXPFGCDN_EL1 value or stops. The value is: **0b1** Error Generation Counter restart mode is implemented and is controlled by ERXPFGCTL_EL1.R. ERXPFGCTL_EL1.R is a read/write field	0b1

Bits	Name	Description	Reset
[29]	SYN	Syndrome. Fault syndrome injection. The value is: **0b0** When an injected error is recorded, the node sets ERR<n>STATUS.{IERR, SERR} to **IMPLEMENTATION DEFINED** values. ERR<n>STATUS.{IERR, SERR} are **UNKNOWN** when ERR<n>STATUS.V is 0.	0b0
[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. **0b0** The component fakes detection of the error on an access to the component.	0b0
[27:13]	**RES0**	Reserved	**RES0**
[12]	MV	Miscellaneous syndrome. Defines whether software can control all or part of the syndrome recorded in the ERR<n>MISC<m> registers when an injected error is recorded. **0b0** When an injected error is recorded, the node might update the ERR<n>MISC<m> registers: • If any syndrome is recorded by the node in the ERR<n>MISC<m> registers, then ERR<n>STATUS.MV is set to 1. • Otherwise, ERR<n>STATUS.MV is unchanged. **Note:** If ERXPFGF_EL1.MV == 0b1, software can write specific values into the ERXMISC<m>_EL1 registers when setting up a fault injection event. The values that can be written to these registers are **IMPLEMENTATION DEFINED**.	0b0
[11]	AV	Address syndrome. Address syndrome injection. The value is: **0b0** When an injected error is recorded, the node might record an address in ERR<n>ADDR. If an address is recorded in ERR<n>ADDR, then ERR<n>STATUS.AV is set to 1. Otherwise, ERR<n>ADDR and ERR<n>STATUS.AV are unchanged.	0b0
[10]	PN	Poison flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.PN status flag. The value is: **0b0** When an injected error is recorded, the node sets ERXSTATUS_EL1.PN to 0.	0b0
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.ER status flag. The value is: **0b0** When an injected error is recorded, the node sets ERR<n>STATUS.ER according to the architecture-defined rules for setting the ER field.	0b0
[8]	CI	Critical Error flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.CI status flag. The value is: **0b0** The node does not support this type of flag This behavior replaces the architecture-defined rules for setting the CI bit.	0b0

Bits	Name	Description	Reset
[7:6]	CE	Corrected Error generation. The value is: **0b01** The fault generation feature of the node allows generation of a non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10. All other values are reserved.	0b01
[5]	DE	Deferred Error generation. The value is: **0b1** The fault generation feature of the node allows generation of this type of error.	0b1
[4]	UEO	Latent or Restartable Error generation. The value is: **0b0** The fault generation feature of the node cannot generate this type of error.	0b0
[3]	UER	Signaled or Recoverable Error generation. The value is: **0b0** The fault generation feature of the node cannot generate this type of error.	0b0
[2]	UEU	Unrecoverable Error generation. The value is: **0b0** The fault generation feature of the node cannot generate this type of error.	0b0
[1]	UC	Uncontainable Error generation. The value is: **0b1** The fault generation feature of the node allows generation of this type of error.	0b1
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.OF status flag. The value is: **0b0** When an injected error is recorded, the node sets ERR<n>STATUS.OF according to the architecture-defined rules for setting the OF field. ERR<n>PFGCTL.OF is **RES0**.	0b0

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**.

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.10.8 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

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 00xx xxxx xxxx xxxx xxx1 x0x0 000x xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-122: AARCH64_ERXPFGCTL_EL1 bit assignments

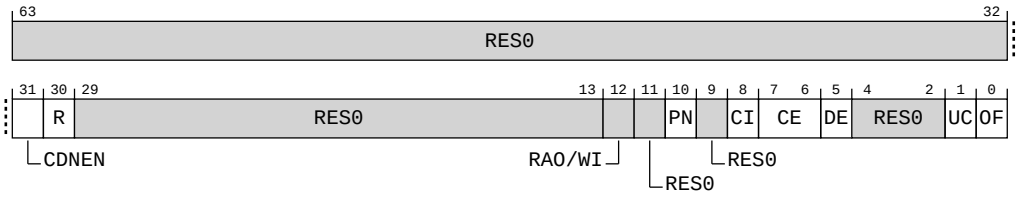


Table A-334: ERXPFGCTL_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	CDNEN	Countdown Enable. Controls transfers from the value that is held in the ERXPFGCDN_EL1 into 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 0b1 to this bit, the Error Generation Counter is set to ERXPFGCDN_EL1.CDN. Cold reset only. Unaffected by Warm reset	0b0

Bits	Name	Description	Reset
[30]	R	Restart. Controls whether, upon reaching zero, the Error Generation Counter restarts from the ERXPFGCDN_EL1 value or stops. **0b0** On reaching 0, the Error Generation Counter will stop. **0b1** On reaching 0, the Error Generation Counter is set to ERXPFGCDN_EL1.CDN. Cold reset only. Unaffected by Warm reset	0b0
[29:13]	**RES0**	Reserved	**RES0**
[12]	**RAO/WI**	Reserved	**RAO/WI**
[11]	**RES0**	Reserved	**RES0**
[10]	PN	Poison flag. The value that is written to ERXSTATUS.PN when an injected error is recorded. **0b00** ERXSTATUS.PN is set to 0 when an injected error is recorded. **0b01** ERXSTATUS.PN is set to 1 when an injected error is recorded.	0b0
[9]	**RES0**	Reserved	**RES0**
[8]	CI	Critical Error flag. The value that is written to ERXSTATUS.CI when an injected error is recorded. **0b0** ERXSTATUS.CI is set to 0 when an injected error is recorded. **0b1** ERXSTATUS.CI is set to 1 when an injected error is recorded.	0b0
[7:6]	CE	Corrected Error generation enable. Controls the type of Corrected Error condition that might be generated. The possible values are: **0b00** No error of this type will be generated. **0b01** A non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10, might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b00
[5]	DE	Deferred Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0
[4:2]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[1]	UC	Uncontainable Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0
[0]	OF	Uncontainable Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0

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**.

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];
```

```
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.10.9 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

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-123: AARCH64_ERXPFGCDN_EL1 bit assignments

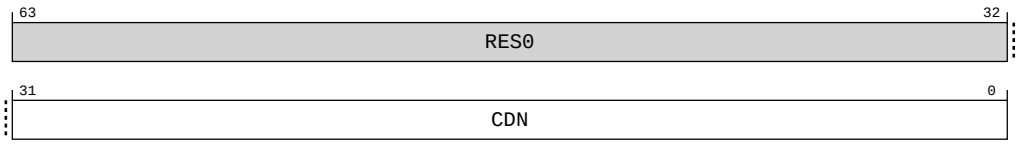


Table A-337: ERXPFGCDN_EL1 bit descriptions

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 ERXPFGCTL_EL1.CDNEN with 1. • The Error Generation Counter decrements to zero and ERXPFGCTL_EL1.R == 0b1. Unaffected by Cold or Warm reset. **Note:** The current Error Generation Counter value is not visible to software.	32{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;
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];
```

A.10.10 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

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-124: AARCH64_ERXMISC0_EL1 bit assignments

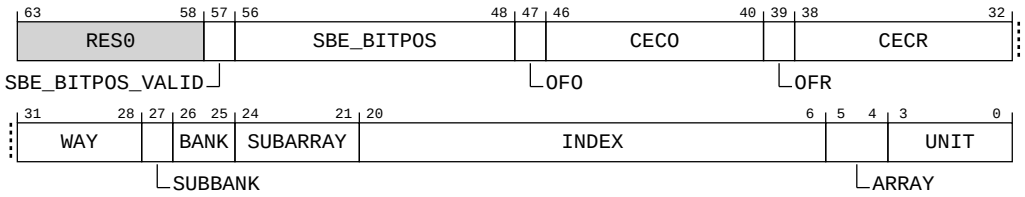


Table A-340: ERXMISC0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:58]	**RES0**	Reserved	**RES0**
[57]	SBE_BITPOS_VALID	Single Bit Error (SBE) bit position field ERXMISC0_EL1.SBE_BITPOS contains valid data **0b0** ERXMISC0_EL1.SBE_BITPOS does not contain valid data. **0b1** ERXMISC0_EL1.SBE_BITPOS contains valid data.	x
[56:48]	SBE_BITPOS	Single Bit Error (SBE) bit position. For a correctable error in a RAM with ECC (L1 data cache, L2 cache), indicates the bit position of the corrected error. Valid when ERXMISC0_EL1.SBE_BITPOS_VALID is 1'b1	9{x}
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERXMISC0_EL1.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 ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Cold or Warm reset.	x
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERXMISC0_EL1.CECR. Unaffected by Cold or Warm reset.	7{x}
[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERXMISC0_EL1.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 ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Cold or Warm reset.	x
[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. This field resets to an **IMPLEMENTATION DEFINED** which might be **UNKNOWN** on a Cold reset. If the reset value is **UNKNOWN**, then the value of this field remains **UNKNOWN** until software initializes it. Unaffected by Cold or Warm reset.	7{x}

Bits	Name	Description	Reset
[31:28]	WAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L1 Data Cache] • Indicates which Tag RAM way or data RAM way detected the error. Upper 2 bits are unused. [L2 TLB] • Indicates which RAM detected an error. The possible values are 0 (RAM 1) to 9 (RAM 10). [L1 Instruction Cache] • Indicates which way detected the error. Upper 2 bits are unused. [L2 Cache] • Indicates which way detected the error. Unaffected by Cold or Warm reset.	xxxx
[27]	SUBBANK	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L1 Instruction Cache] • Indicates which subbank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero. Unaffected by Cold or Warm reset.	x
[26:25]	BANK	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which L2 bank detected the error. [L1 Instruction Cache] • Indicates which bank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero. Unaffected by Cold or Warm reset.	xx
[24:21]	SUBARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which L2 data ECC granule detected the error. Upper bits are unused dependent on the ECC granule size. [L1 Data Cache] • Indicates for L1 Data RAM which word had the error detected. For L1 Tag RAMs which bank had the error (0b0000: bank0 , 0b0001: bank1) Unaffected by Cold or Warm reset.	xxxx

Bits	Name	Description	Reset
[20:6]	INDEX	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. [L1 Data Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size [L2 TLB] • Index of TLB RAM. Upper 4 bits are unused. [L1 Instruction Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. Unaffected by Cold or Warm reset.	`15{x}`

Bits	Name	Description	Reset
[5:4]	ARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] Indicates which array detected the error. The possible values are: • `0b00` L2 Tag RAM. • `0b01` L2 Data RAM. • `0b10` L2 TQ Data RAM. • `0b11` CHI Error. [L1 Data Cache] Indicates which array detected the error. The possible values are: • 00 LS Tag RAM 0. • 01 LS Tag RAM 1. • 10 LS Data RAM. • 11 LS Tag RAM 2. [L2 TLB] Indicates which array detected the error. The possible values are: • 00 Translation cache. • 01 GPT cache (when LEGACY_TZ_EN is 0). • 10 Reserved. • 11 Reserved. [L1 Instruction Cache] Indicates which array that detected the error, Data Array has higher priority. The possible values are: • `0b00` Tag. • `0b01` Data. Unaffected by Cold or Warm reset.	`xx`
[3:0]	UNIT	Indicates the unit which detected the error. The possible values are: **`0b0001`** L1 Instruction Cache. **`0b0010`** L2 TLB. **`0b0100`** L1 Data Cache. **`0b1000`** L2 Cache.	`xxxx`

Access

`MRS <Xt>, ERXMISC0_EL1`

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

`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);
    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.10.11 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

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-125: AARCH64_ERXMISC1_EL1 bit assignments

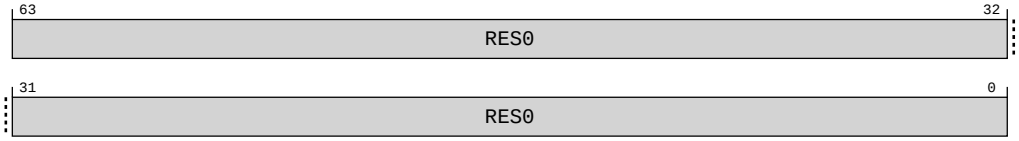


Table A-343: ERXMISC1_EL1 bit descriptions

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

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
            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.10.12 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

Functional group

RAS

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-126: AARCH64_ERXMISC2_EL1 bit assignments

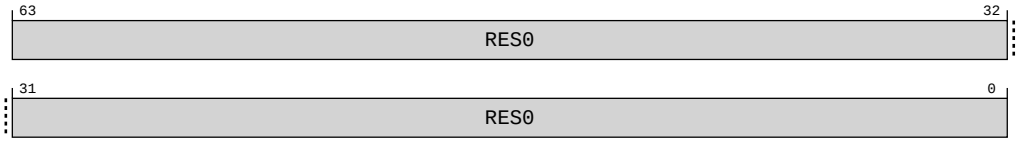


Table A-346: ERXMISC2_EL1 bit descriptions

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

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**.
- 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);
    else
        ERXMISC2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXMISC2_EL1 = X[t, 64];
```

A.10.13 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

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-127: AARCH64_ERXMISC3_EL1 bit assignments

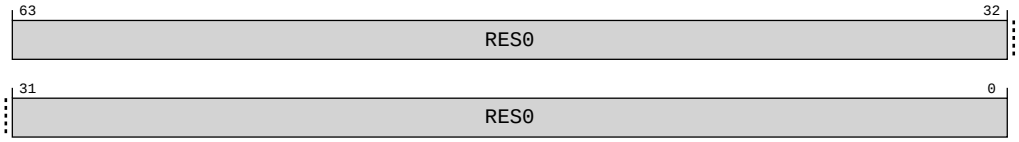


Table A-349: ERXMISC3_EL1 bit descriptions

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

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;
```

```
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.11 AArch64 SPE registers summary

The following summary table provides an overview of all SPE registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-352: SPE registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMSCR_EL1	3	0	C9	C9	0	See individual bit resets.	64-bit	Statistical Profiling Control Register (EL1)
PMSNEVFR_EL1	3	0	C9	C9	1	See individual bit resets.	64-bit	Sampling Inverted Event Filter Register
PMSICR_EL1	3	0	C9	C9	2	See individual bit resets.	64-bit	Sampling Interval Counter Register
PMSIRR_EL1	3	0	C9	C9	3	See individual bit resets.	64-bit	Sampling Interval Reload Register
PMSFCR_EL1	3	0	C9	C9	4	See individual bit resets.	64-bit	Sampling Filter Control Register
PMSEVFR_EL1	3	0	C9	C9	5	See individual bit resets.	64-bit	Sampling Event Filter Register
PMSLATFR_EL1	3	0	C9	C9	6	See individual bit resets.	64-bit	Sampling Latency Filter Register
PMSIDR_EL1	3	0	C9	C9	7	See individual bit resets.	64-bit	Sampling Profiling ID Register
PMBLIMITR_EL1	3	0	C9	C10	0	See individual bit resets.	64-bit	Profiling Buffer Limit Address Register
PMBPTR_EL1	3	0	C9	C10	1	See individual bit resets.	64-bit	Profiling Buffer Write Pointer Register
PMBSR_EL1	3	0	C9	C10	3	See individual bit resets.	64-bit	Profiling Buffer Status/syndrome Register (EL1)
PMBIDR_EL1	3	0	C9	C10	7	0x0000000000000226	64-bit	Profiling Buffer ID Register
PMSCR_EL2	3	4	C9	C9	0	See individual bit resets.	64-bit	Statistical Profiling Control Register (EL2)

A.11.1 PMSNEVFR_EL1, Sampling Inverted Event Filter Register

Controls sample filtering by events. The overall inverted filter is the logical OR of these filters. For example, if PMSNEVFR_EL1.E[3] and PMSNEVFR_EL1.E[5] are both set to 1, samples that have either event 3 (Level 1 unified or data cache refill) or event 5 (TLB walk) set to 1 are not recorded.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

SPE

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx 0000 0000 0000 0000 xxxx xxxx 0000 0xx0 xxxx xxxx 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-128: AARCH64_PMSNEVFR_EL1 bit assignments

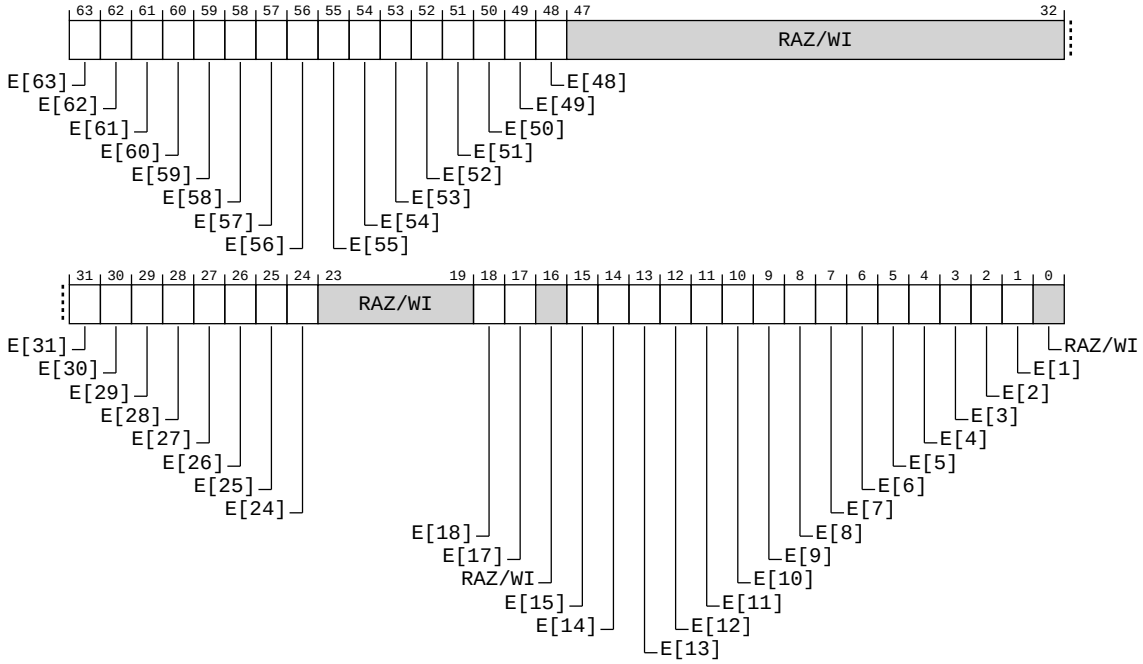


Table A-353: PMSNEVFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	E[63]	**When event 63 is implemented and filtering on event 63 is supported** Filter on **IMPLEMENTATION DEFINED** event 63. **0b0** Event 63 is ignored. **0b1** Do not record samples that have event 63 == 1. **Otherwise** RAZ/WI	xxxx

Bits	Name	Description	Reset
[62]	E[62]	**When event 62 is implemented and filtering on event 62 is supported** Filter on **IMPLEMENTATION DEFINED** event 62. **0b0** Event 62 is ignored. **0b1** Do not record samples that have event 62 == 1. **Otherwise** RAZ/WI	xxxx
[61]	E[61]	**When event 61 is implemented and filtering on event 61 is supported** Filter on **IMPLEMENTATION DEFINED** event 61. **0b0** Event 61 is ignored. **0b1** Do not record samples that have event 61 == 1. **Otherwise** RAZ/WI	xxxx
[60]	E[60]	**When event 60 is implemented and filtering on event 60 is supported** Filter on **IMPLEMENTATION DEFINED** event 60. **0b0** Event 60 is ignored. **0b1** Do not record samples that have event 60 == 1. **Otherwise** RAZ/WI	xxxx
[59]	E[59]	**When event 59 is implemented and filtering on event 59 is supported** Filter on **IMPLEMENTATION DEFINED** event 59. **0b0** Event 59 is ignored. **0b1** Do not record samples that have event 59 == 1. **Otherwise** RAZ/WI	xxxx

Bits	Name	Description	Reset
[58]	E[58]	**When event 58 is implemented and filtering on event 58 is supported** Filter on **IMPLEMENTATION DEFINED** event 58. **0b0** Event 58 is ignored. **0b1** Do not record samples that have event 58 == 1. **Otherwise** RAZ/WI	xxxx
[57]	E[57]	**When event 57 is implemented and filtering on event 57 is supported** Filter on **IMPLEMENTATION DEFINED** event 57. **0b0** Event 57 is ignored. **0b1** Do not record samples that have event 57 == 1. **Otherwise** RAZ/WI	xxxx
[56]	E[56]	**When event 56 is implemented and filtering on event 56 is supported** Filter on **IMPLEMENTATION DEFINED** event 56. **0b0** Event 56 is ignored. **0b1** Do not record samples that have event 56 == 1. **Otherwise** RAZ/WI	xxxx
[55]	E[55]	**When event 55 is implemented and filtering on event 55 is supported** Filter on **IMPLEMENTATION DEFINED** event 55. **0b0** Event 55 is ignored. **0b1** Do not record samples that have event 55 == 1. **Otherwise** RAZ/WI	xxxx

Bits	Name	Description	Reset
[54]	E[54]	**When event 54 is implemented and filtering on event 54 is supported** Filter on **IMPLEMENTATION DEFINED** event 54. **0b0** Event 54 is ignored. **0b1** Do not record samples that have event 54 == 1. **Otherwise** RAZ/WI	xxxx
[53]	E[53]	**When event 53 is implemented and filtering on event 53 is supported** Filter on **IMPLEMENTATION DEFINED** event 53. **0b0** Event 53 is ignored. **0b1** Do not record samples that have event 53 == 1. **Otherwise** RAZ/WI	xxxx
[52]	E[52]	**When event 52 is implemented and filtering on event 52 is supported** Filter on **IMPLEMENTATION DEFINED** event 52. **0b0** Event 52 is ignored. **0b1** Do not record samples that have event 52 == 1. **Otherwise** RAZ/WI	xxxx
[51]	E[51]	**When event 51 is implemented and filtering on event 51 is supported** Filter on **IMPLEMENTATION DEFINED** event 51. **0b0** Event 51 is ignored. **0b1** Do not record samples that have event 51 == 1. **Otherwise** RAZ/WI	xxxx

Bits	Name	Description	Reset
[50]	E[50]	**When event 50 is implemented and filtering on event 50 is supported** Filter on **IMPLEMENTATION DEFINED** event 50. **0b0** Event 50 is ignored. **0b1** Do not record samples that have event 50 == 1. **Otherwise** RAZ/WI	xxxx
[49]	E[49]	**When event 49 is implemented and filtering on event 49 is supported** Filter on **IMPLEMENTATION DEFINED** event 49. **0b0** Event 49 is ignored. **0b1** Do not record samples that have event 49 == 1. **Otherwise** RAZ/WI	xxxx
[48]	E[48]	**When event 48 is implemented and filtering on event 48 is supported** Filter on **IMPLEMENTATION DEFINED** event 48. **0b0** Event 48 is ignored. **0b1** Do not record samples that have event 48 == 1. **Otherwise** RAZ/WI	xxxx
[47:32]	**RAZ/WI**	Reserved	**RAZ/WI**
[31]	E[31]	**When event 31 is implemented and filtering on event 31 is supported** Filter on **IMPLEMENTATION DEFINED** event 31. **0b0** Event 31 is ignored. **0b1** Do not record samples that have event 31 == 1. **Otherwise** RAZ/WI	xxxx

Bits	Name	Description	Reset
[30]	E[30]	**When event 30 is implemented and filtering on event 30 is supported** Filter on **IMPLEMENTATION DEFINED** event 30. **0b0** Event 30 is ignored. **0b1** Do not record samples that have event 30 == 1. **Otherwise** RAZ/WI	xxxx
[29]	E[29]	**When event 29 is implemented and filtering on event 29 is supported** Filter on **IMPLEMENTATION DEFINED** event 29. **0b0** Event 29 is ignored. **0b1** Do not record samples that have event 29 == 1. **Otherwise** RAZ/WI	xxxx
[28]	E[28]	**When event 28 is implemented and filtering on event 28 is supported** Filter on **IMPLEMENTATION DEFINED** event 28. **0b0** Event 28 is ignored. **0b1** Do not record samples that have event 28 == 1. **Otherwise** RAZ/WI	xxxx
[27]	E[27]	**When event 27 is implemented and filtering on event 27 is supported** Filter on **IMPLEMENTATION DEFINED** event 27. **0b0** Event 27 is ignored. **0b1** Do not record samples that have event 27 == 1. **Otherwise** RAZ/WI	xxxx

Bits	Name	Description	Reset
[26]	E[26]	**When event 26 is implemented and filtering on event 26 is supported** Filter on **IMPLEMENTATION DEFINED** event 26. **0b0** Event 26 is ignored. **0b1** Do not record samples that have event 26 == 1. **Otherwise** RAZ/WI	xxxx
[25]	E[25]	**When event 25 is implemented and filtering on event 25 is supported** Filter on **IMPLEMENTATION DEFINED** event 25. **0b0** Event 25 is ignored. **0b1** Do not record samples that have event 25 == 1. **Otherwise** RAZ/WI	xxxx
[24]	E[24]	**When event 24 is implemented and filtering on event 24 is supported** Filter on **IMPLEMENTATION DEFINED** event 24. **0b0** Event 24 is ignored. **0b1** Do not record samples that have event 24 == 1. **Otherwise** RAZ/WI	xxxx
[23:19]	**RAZ/WI**	Reserved	**RAZ/WI**
[18]	E[18]	Filter on Not empty predicate event. **0b0** Empty predicate event is ignored. **0b1** Do not record samples that have the Empty predicate event == 1.	x
[17]	E[17]	Filter on Not partial predicate event. **0b0** Partial or empty predicate event is ignored. **0b1** Do not record samples that have the Partial or empty predicate event == 1.	x
[16]	**RAZ/WI**	Reserved	**RAZ/WI**

Bits	Name	Description	Reset
[15]	E[15]	**When event 15 is implemented and filtering on event 15 is supported** Filter on **IMPLEMENTATION DEFINED** event 15. **0b0** Event 15 is ignored. **0b1** Do not record samples that have event 15 == 1. **Otherwise** RAZ/WI	xxxx
[14]	E[14]	**When event 14 is implemented and filtering on event 14 is supported** Filter on **IMPLEMENTATION DEFINED** event 14. **0b0** Event 14 is ignored. **0b1** Do not record samples that have event 14 == 1. **Otherwise** RAZ/WI	xxxx
[13]	E[13]	**When event 13 is implemented and filtering on event 13 is supported** Filter on **IMPLEMENTATION DEFINED** event 13. **0b0** Event 13 is ignored. **0b1** Do not record samples that have event 13 == 1. **Otherwise** RAZ/WI	xxxx
[12]	E[12]	**When event 12 is implemented and filtering on event 12 is supported** Filter on **IMPLEMENTATION DEFINED** event 12. **0b0** Event 12 is ignored. **0b1** Do not record samples that have event 12 == 1. **Otherwise** RAZ/WI	xxxx
[11]	E[11]	Filter on Aligned event. **0b0** Misalignment event is ignored. **0b1** Do not record samples that have the Misalignment event == 1.	x

Bits	Name	Description	Reset
[10]	E[10]	**When filtering on event 10 is optionally supported and event 10 is implemented** Filter on No remote access event. **0b0** Remote access event is ignored. **0b1** Do not record samples that have the Remote access event == 1. **Otherwise** RAZ/WI	xxxx
[9]	E[9]	**When filtering on event 9 is optionally supported and event 9 is implemented** Filter on Last Level cache hit event. **0b0** Last Level cache miss event is ignored. **0b1** Do not record samples that have the Last Level cache miss event == 1. **Otherwise** RAZ/WI	xxxx
[8]	E[8]	**When filtering on event 8 is optionally supported and event 8 is implemented** Filter on No Last Level cache access event. **0b0** Last Level cache access event is ignored. **0b1** Do not record samples that have the Last Level cache access event == 1. **Otherwise** RAZ/WI	xxxx
[7]	E[7]	Filter on Correctly predicted event. **0b0** Mispredicted event is ignored. **0b1** Do not record samples that have the Mispredicted event == 1.	x
[6]	E[6]	Filter on Taken event. **0b0** Not taken event is ignored. **0b1** Do not record samples that have the Not taken event == 1.	x
[5]	E[5]	Filter on TLB hit event. **0b0** TLB walk event is ignored. **0b1** Do not record samples that have the TLB walk event == 1.	x

Bits	Name	Description	Reset
[4]	E[4]	**When filtering on event 4 is optionally supported** Filter on No TLB access event. **0b0** TLB access event is ignored. **0b1** Do not record samples that have the TLB access event == 1. **Otherwise** RAZ/WI	xxxx
[3]	E[3]	Filter on Level 1 data cache hit event. **0b0** Level 1 data cache refill or miss event is ignored. **0b1** Do not record samples that have the Level 1 data cache refill or miss event == 1.	x
[2]	E[2]	**When filtering on event 2 is optionally supported** Filter on No Level 1 data cache access event. **0b0** Level 1 data cache access event is ignored. **0b1** Do not record samples that have the Level 1 data cache access event == 1. **Otherwise** RAZ/WI	xxxx
[1]	E[1]	**When ImpDefBool("the PE supports sampling of speculative instructions** Filter on Speculative event. **0b0** Architecturally retired event is ignored. **0b1** Do not record samples that have the Architecturally retired event == 1. **Otherwise** RAZ/WI	xxxx
[0]	**RAZ/WI**	Reserved	**RAZ/WI**

Access

MRS <Xt>, PMSNEVFR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b001

MSR PMSNEVFR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b001

Accessibility

MRS <Xt>, PMSNEVFR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] !=
 SCR_EL3.NS) then
        UNDEFINED;
    elsif EL3SDDUndefPriority() && MDCR_EL3.EnPMSN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nPMSNEVFR_EL1 == '0'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.EnPMSN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'1x1'} then
        X[t, 64] = NVMem[0x850];
    else
        X[t, 64] = PMSNEVFR_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] !=
 SCR_EL3.NS) then
        UNDEFINED;
    elsif EL3SDDUndefPriority() && MDCR_EL3.EnPMSN == '0' then
        UNDEFINED;
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.EnPMSN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSNEVFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSNEVFR_EL1;
```

MSR PMSNEVFR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] !=
 SCR_EL3.NS) then
        UNDEFINED;
    elsif EL3SDDUndefPriority() && MDCR_EL3.EnPMSN == '0' then
        UNDEFINED;
```

```
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.nPMSNEVFR_EL1 == '0'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.EnPMSN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EffectiveHCR_EL2_NVx() IN {'1x1'} then
        NVMem[0x850] = X[t, 64];
    else
        PMSNEVFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] !=
 SCR_EL3.NS) then
        UNDEFINED;
    elsif EL3SDDUndefPriority() && MDCR_EL3.EnPMSN == '0' then
        UNDEFINED;
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.EnPMSN == '0' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSNEVFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSNEVFR_EL1 = X[t, 64];
```

A.11.2 PMSEVFR_EL1, Sampling Event Filter Register

Controls sample filtering by events. The overall filter is the logical AND of these filters. For example, if PMSEVFR_EL1.E[3] and PMSEVFR_EL1.E[5] are both set to 1, only samples that have both event 3 (Level 1 unified or data cache refill) and event 5 (TLB walk) set to 1 are recorded.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

SPE

Access type

RW

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0xx0 0000 x000 xxx0 x0x0
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-129: AARCH64_PMSEVFR_EL1 bit assignments

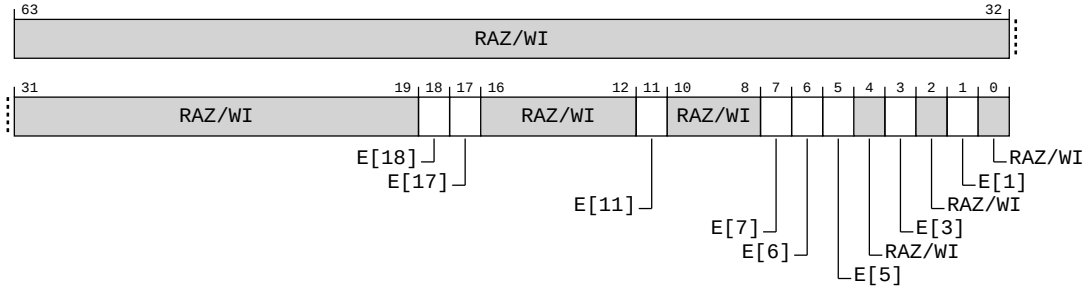


Table A-356: PMSEVFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RAZ/WI**	Reserved	**RAZ/WI**
[18]	E[18]	Filter on Empty predicate event. **0b0** Empty predicate event is ignored. **0b1** Do not record samples that have the Empty predicate event == 0.	x
[17]	E[17]	Filter on Partial or empty predicate event. **0b0** Partial or empty predicate event is ignored. **0b1** Do not record samples that have the Partial or empty predicate event == 0.	x
[16:12]	**RAZ/WI**	Reserved	**RAZ/WI**
[11]	E[11]	Filter on Misalignment event. **0b0** Misalignment event is ignored. **0b1** Do not record samples that have the Misalignment event == 0.	x
[10:8]	**RAZ/WI**	Reserved	**RAZ/WI**

Bits	Name	Description	Reset
[7]	E[7]	Filter on Mispredicted event. **0b0** Mispredicted event is ignored. **0b1** Do not record samples that have the Mispredicted event == 0.	x
[6]	E[6]	Filter on Not taken event. **0b0** Not taken event is ignored. **0b1** Do not record samples that have the Not taken event == 0.	x
[5]	E[5]	Filter on TLB walk event. **0b0** TLB walk event is ignored. **0b1** Do not record samples that have the TLB walk event == 0.	x
[4]	**RAZ/WI**	Reserved	**RAZ/WI**
[3]	E[3]	Filter on Level 1 data cache refill or miss event. **0b0** Level 1 data cache refill or miss event is ignored. **0b1** Do not record samples that have the Level 1 data cache refill or miss event == 0.	x
[2]	**RAZ/WI**	Reserved	**RAZ/WI**
[1]	E[1]	Filter on Architecturally retired event. **0b0** Architecturally retired event is ignored. **0b1** Do not record samples that have the Architecturally retired event == 0.	x
[0]	**RAZ/WI**	Reserved	**RAZ/WI**

Access

```
MRS <Xt>, PMSEVFR_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b101

```
MSR PMSEVFR_EL1, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b101

Accessibility

MRS <Xt>, PMSEVFR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] !=
 SCR_EL3.NS) then
        UNDEFINED;
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMSEVFR_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSEVFR_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] !=
 SCR_EL3.NS) then
        UNDEFINED;
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSEVFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSEVFR_EL1;
```

MSR PMSEVFR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] !=
 SCR_EL3.NS) then
        UNDEFINED;
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMSEVFR_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSEVFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] !=
 SCR_EL3.NS) then
        UNDEFINED;
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSEVFR_EL1 = X[t, 64];
```

```
elsif PSTATE.EL == EL3 then
    PMSEVFR_EL1 = X[t, 64];
```

A.11.3 PMSIDR_EL1, Sampling Profiling ID Register

Describes the Statistical Profiling implementation to software.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

SPE

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 000x 0000 0011 0110 0100 0101 0111
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-130: AARCH64_PMSIDR_EL1 bit assignments

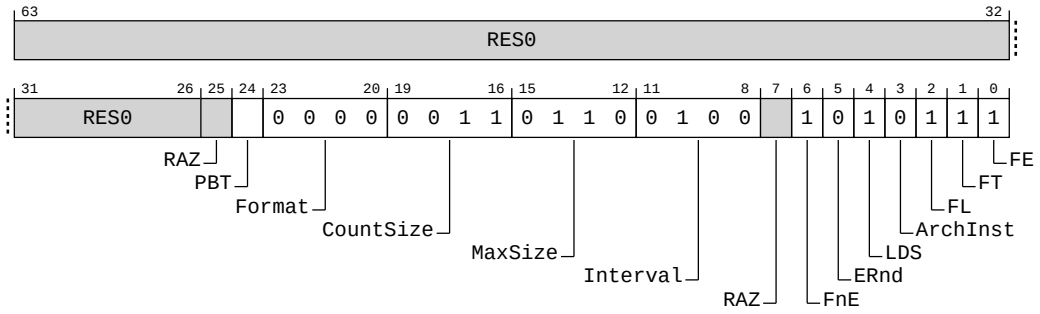


Table A-359: PMSIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:26]	**RES0**	Reserved	**RES0**
[25]	**RAZ**	Reserved	**RAZ**
[24]	PBT	Previous branch target Address packet. **0b0** Previous branch target Address packet not supported. **0b1** Previous branch target Address packet support implemented.	The reset values can be the following: `0b0`, `0b1`, respective to the value.
[23:20]	Format	Defines the format of the sample records. **0b0000** Format 0.	`0b0000`
[19:16]	CountSize	Defines the size of the counters. **0b0011** 16-bit saturating counters.	`0b0011`
[15:12]	MaxSize	Defines the largest size for a single record, rounded up to a power-of-two. If this is the same as the minimum alignment (PMBIDR_EL1.Align), then each record is exactly this size. **0b0110** 64 bytes.	`0b0110`
[11:8]	Interval	Recommended minimum sampling interval. This provides guidance from the implementer to the smallest minimum sampling interval, N. **0b0100** 1,024.	`0b0100`
[7]	**RAZ**	Reserved	**RAZ**
[6]	FnE	Filtering by events, inverted. **0b1** PMSNEVFR_EL1 and PMSFCR_EL1.FnE are implemented.	`0b1`
[5]	ERnd	Defines how the random number generator is used in determining the interval between samples, when enabled by PMSIRR_EL1.RND. **0b0** The random number is added at the start of the interval, and the sample is taken and a new interval started when the combined interval expires.	`0b0`
[4]	LDS	Data source indicator for sampled load instructions. **0b1** Loaded data source implemented.	`0b1`
[3]	ArchInst	Architectural instruction profiling. **0b0** Micro-op sampling implemented.	`0b0`
[2]	FL	Filtering by latency. **0b1**	`0b1`
[1]	FT	Filtering by operation type. **0b1**	`0b1`

Bits	Name	Description	Reset
[0]	FE	Filtering by events. **0b1**	0b1

Access

MRS <Xt>, PMSIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b111

Accessibility

MRS <Xt>, PMSIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] !=
 SCR_EL3.NS) then
        UNDEFINED;
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMSIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSIDR_EL1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] !=
 SCR_EL3.NS) then
        UNDEFINED;
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSIDR_EL1;
```

A.11.4 PMBIDR_EL1, Profiling Buffer ID Register

Provides information to software as to whether the buffer can be programmed at the current Exception level.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

SPE

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010 0010 0110
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-131: AARCH64_PMBIDR_EL1 bit assignments

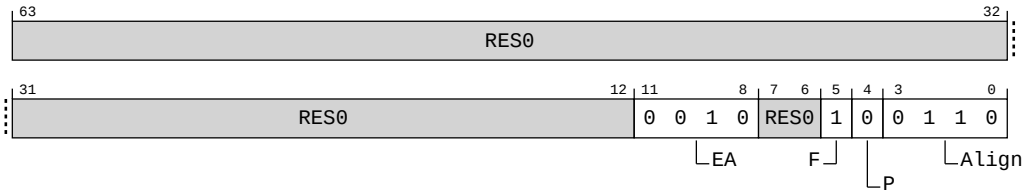


Table A-361: PMBIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:12]	**RES0**	Reserved	**RES0**
[11:8]	EA	External Abort handling. Describes how the PE manages External aborts on writes made by the Statistical Profiling Unit to the Profiling Buffer. **0b0010** An External abort on a write made by the Statistical Profiling Unit generates an asynchronous SError exception at the PE.	0b0010
[7:6]	**RES0**	Reserved	**RES0**
[5]	F	Flag updates. Describes how address translations performed by the Statistical Profiling Unit manage the Access flag and dirty state. **0b1** Hardware management of the Access flag and dirty state for accesses made by the Statistical Profiling Unit is controlled in the same way as explicit memory accesses in the Profiling Buffer owning translation regime.	0b1
[4]	P	Programming not allowed. When read at EL3, this field reads as zero. Otherwise, indicates that the Profiling Buffer is owned by a higher Exception level or another Security state. **0b0** Programming is allowed.	0b0
[3:0]	Align	Defines the minimum alignment constraint for writes to PMBPTR_EL1. **0b0110** 64 bytes.	0b0110

Access

MRS <Xt>, PMBIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b111

Accessibility

MRS <Xt>, PMBIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMBIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMBIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMBIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMBIDR_EL1;
```

A.12 AArch64 Special registers summary

The following summary table provides an overview of all Special registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-363: Special registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SPSR_EL1	3	0	C4	C0	0	See individual bit resets.	64-bit	Saved Program Status Register (EL1)
ELR_EL1	3	0	C4	C0	1	See individual bit resets.	64-bit	Exception Link Register (EL1)
SP_EL0	3	0	C4	C1	0	See individual bit resets.	64-bit	Stack Pointer (EL0)
DSPSR_EL0	3	3	C4	C5	0	See individual bit resets.	64-bit	Debug Saved Program Status Register
DLR_EL0	3	3	C4	C5	1	See individual bit resets.	64-bit	Debug Link Register
SPSR_EL2	3	4	C4	C0	0	See individual bit resets.	64-bit	Saved Program Status Register (EL2)
ELR_EL2	3	4	C4	C0	1	See individual bit resets.	64-bit	Exception Link Register (EL2)
SP_EL1	3	4	C4	C1	0	See individual bit resets.	64-bit	Stack Pointer (EL1)
SPSR_irq	3	4	C4	C3	0	See individual bit resets.	64-bit	Saved Program Status Register (IRQ mode)
SPSR_abt	3	4	C4	C3	1	See individual bit resets.	64-bit	Saved Program Status Register (Abort mode)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SPSR_und	3	4	C4	C3	2	See individual bit resets.	64-bit	Saved Program Status Register (Undefined mode)
SPSR_fiq	3	4	C4	C3	3	See individual bit resets.	64-bit	Saved Program Status Register (FIQ mode)
SPSR_EL3	3	6	C4	C0	0	See individual bit resets.	64-bit	Saved Program Status Register (EL3)
ELR_EL3	3	6	C4	C0	1	See individual bit resets.	64-bit	Exception Link Register (EL3)
SP_EL2	3	6	C4	C1	0	See individual bit resets.	64-bit	Stack Pointer (EL2)

A.13 AArch64 System Instructions summary

The following summary table provides an overview of all 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-364: System Instructions summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
COSP_RCTX	1	3	C7	C3	6	See individual bit resets.	64-bit	Clear Other Speculative Prediction Restriction by Context
SYS_IMP_RAMINDEX	1	6	C15	C0	0	See individual bit resets.	64-bit	RAM Index

A.13.1 COSP RCTX, Clear Other Speculative Prediction Restriction by Context

Clear Other Speculative Prediction Restriction by Context applies to all prediction resources not managed by other speculation restriction System instructions.

The actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control any predictions occurring after the instruction is complete and synchronized.

This instruction applies to all speculative access except:

- Cache Prefetch predictions.
- Control Flow predictions.
- Data Value predictions.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the PE that executed the original restriction instruction, and a subsequent Context Synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

Note

This instruction does not require the invalidation of Cache Allocation Resources so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations, the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System Instructions

Bit descriptions

Figure A-132: AARCH64_COSP_RCTX bit assignments

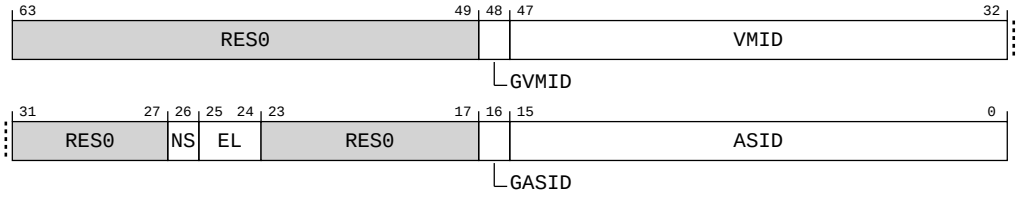


Table A-365: COSP RCTX bit descriptions

Bits	Name	Description	Reset
[63:49]	**RES0**	Reserved	**RES0**
[48]	GVMID	Execution of this instruction applies to all VMIDs or a specified VMID. **0b0** Applies to specified VMID for an EL0 or EL1 target execution context. **0b1** Applies to all VMIDs for an EL0 or EL1 target execution context.	x

Bits	Name	Description	Reset
[47:32]	VMID	Only applies when bit[48] is 0 and the target execution context is either: • EL1. • EL0 when the Effective value of HCR_EL2.{E2H, TGE} is not {1, 1}. Otherwise this field is **RES0**. When the instruction is executed at EL1, this field is treated as the current VMID. When the instruction is executed at EL0 and the Effective value of HCR_EL2.{E2H, TGE} is not {1, 1}, this field is treated as the current VMID. When the instruction is executed at EL0 and the Effective value of HCR_EL2.{E2H, TGE} is {1, 1}, this field is ignored. If EL2 is not implemented or not enabled for the target Security state, this field is **RES0**. If the implementation supports 16 bits of VMID, then the upper 8 bits of the VMID must be written to 0 by software when the context being affected only uses 8 bits.	16{x}
[31:27]	**RES0**	Reserved	**RES0**
[26]	NS	Security State. Defined values are: **0b0** Secure state. **0b1** Non-secure state.	x
[25:24]	EL	Exception Level. Indicates the Exception level of the target execution context. **0b00** EL0. **0b01** EL1. **0b10** EL2. **0b11** EL3.	xx
[23:17]	**RES0**	Reserved	**RES0**
[16]	GASID	Execution of this instruction applies to all ASIDs, or a specified ASID. **0b0** Applies to specified ASID for an EL0 target execution context. **0b1** Applies to all ASIDs for an EL0 target execution context.	x
[15:0]	ASID	Only applies to an EL0 target execution context and when bit[16] is 0. Otherwise, this field is **RES0**. When the instruction is executed at EL0, this field is treated as the current ASID. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being affected only uses 8 bits.	16{x}

Access

COSP RCTX, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0011	0b110

Accessibility

COSP RCTX, <Xt>

```
if PSTATE.EL == EL0 then
    if !ELIsInHost(EL0) && SCTLR_EL1.EnRCTX == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && !ELIsInHost(EL0) && SCR_EL3.FGTEn == '1' &&
 HFGITR_EL2.COSPRCTX == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ELIsInHost(EL0) && SCTLR_EL2.EnRCTX == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.RestrictPrediction(X[t, 64], RestrictType_Other);
elsif PSTATE.EL == EL1 then
    if EffectiveHCR_EL2_NVx() IN {'xx1'} then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGITR_EL2.COSPRCTX == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.RestrictPrediction(X[t, 64], RestrictType_Other);
elsif PSTATE.EL == EL2 then
    AArch64.RestrictPrediction(X[t, 64], RestrictType_Other);
elsif PSTATE.EL == EL3 then
    AArch64.RestrictPrediction(X[t, 64], RestrictType_Other);
```

A.13.2 SYS_IMP_RAMINDEX, RAM Index

Read contents of the CPU cache specified by the source register into IMP_IDATA0_EL3, IMP_IDATA1_EL3, IMP_IDATA2_EL3, IMP_DDATA0_EL3, IMP_DDATA1_EL3, and IMP_DDATA2_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System Instructions

Bit descriptions

Figure A-133: AARCH64_SYS_IMP_RAMINDEX bit assignments

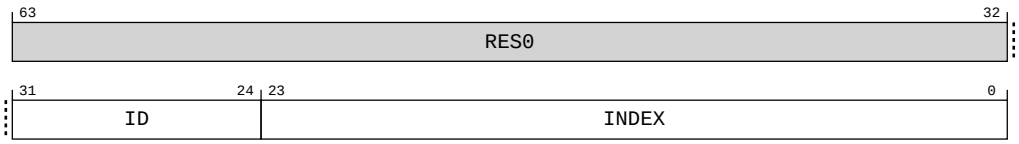


Table A-367: SYS_IMP_RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:24]	ID	RAM ID (See Chapter 10)	8{x}
[23:0]	INDEX	RAM Index (See Chapter 10)	24{x}

Access

```
SYS #6, C15, C0, #0{, <Xt>}
```

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0000	0b000

Accessibility

```
SYS #6, C15, C0, #0{, <Xt>}
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_RAMINDEX(X[t, 64]);
```

A.14 AArch64 TRBE registers summary

The following summary table provides an overview of all TRBE registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-369: TRBE registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRBLIMITR_EL1	3	0	C9	C11	0	See individual bit resets.	64-bit	Trace Buffer Limit Address Register
TRBPTR_EL1	3	0	C9	C11	1	See individual bit resets.	64-bit	Trace Buffer Write Pointer Register
TRBBASER_EL1	3	0	C9	C11	2	See individual bit resets.	64-bit	Trace Buffer Base Address Register
TRBSR_EL1	3	0	C9	C11	3	See individual bit resets.	64-bit	Trace Buffer Status/syndrome Register (EL1)
TRBMAR_EL1	3	0	C9	C11	4	See individual bit resets.	64-bit	Trace Buffer Memory Attribute Register
TRBTRG_EL1	3	0	C9	C11	6	See individual bit resets.	64-bit	Trace Buffer Trigger Counter Register
TRBIDR_EL1	3	0	C9	C11	7	See individual bit resets.	64-bit	Trace Buffer ID Register

A.15 AArch64 Timer registers summary

The following summary table provides an overview of all Timer registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-370: Timer registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
CNTKCTL_EL1	3	0	C14	C1	0	See individual bit resets.	64-bit	Counter-timer Kernel Control Register
CNTFRQ_EL0	3	3	C14	C0	0	See individual bit resets.	64-bit	Counter-timer Frequency Register
CNTPCT_EL0	3	3	C14	C0	1	See individual bit resets.	64-bit	Counter-timer Physical Count Register
CNTVCT_EL0	3	3	C14	C0	2	See individual bit resets.	64-bit	Counter-timer Virtual Count Register
CNTPCTSS_EL0	3	3	C14	C0	5	See individual bit resets.	64-bit	Counter-timer Self-Synchronized Physical Count Register
CNTVCTSS_EL0	3	3	C14	C0	6	See individual bit resets.	64-bit	Counter-timer Self-Synchronized Virtual Count Register
CNTP_TVAL_EL0	3	3	C14	C2	0	See individual bit resets.	64-bit	Counter-timer Physical Timer TimerValue Register
CNTP_CTL_EL0	3	3	C14	C2	1	See individual bit resets.	64-bit	Counter-timer Physical Timer Control Register
CNTP_CVAL_EL0	3	3	C14	C2	2	See individual bit resets.	64-bit	Counter-timer Physical Timer CompareValue Register
CNTV_TVAL_EL0	3	3	C14	C3	0	See individual bit resets.	64-bit	Counter-timer Virtual Timer TimerValue Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
CNTV_CTL_EL0	3	3	C14	C3	1	See individual bit resets.	64-bit	Counter-timer Virtual Timer Control Register
CNTV_CVAL_EL0	3	3	C14	C3	2	See individual bit resets.	64-bit	Counter-timer Virtual Timer CompareValue Register
CNTVOFF_EL2	3	4	C14	C0	3	See individual bit resets.	64-bit	Counter-timer Virtual Offset Register
CNTPOFF_EL2	3	4	C14	C0	6	See individual bit resets.	64-bit	Counter-timer Physical Offset Register
CNTHCTL_EL2	3	4	C14	C1	0	See individual bit resets.	64-bit	Counter-timer Hypervisor Control Register
CNTHP_TVAL_EL2	3	4	C14	C2	0	See individual bit resets.	64-bit	Counter-timer Physical Timer TimerValue Register (EL2)
CNTHP_CTL_EL2	3	4	C14	C2	1	See individual bit resets.	64-bit	Counter-timer Hypervisor Physical Timer Control Register
CNTHP_CVAL_EL2	3	4	C14	C2	2	See individual bit resets.	64-bit	Counter-timer Physical Timer CompareValue Register (EL2)
CNTHV_TVAL_EL2	3	4	C14	C3	0	See individual bit resets.	64-bit	Counter-timer Virtual Timer TimerValue Register (EL2)
CNTHV_CTL_EL2	3	4	C14	C3	1	See individual bit resets.	64-bit	Counter-timer Virtual Timer Control Register (EL2)
CNTHV_CVAL_EL2	3	4	C14	C3	2	See individual bit resets.	64-bit	Counter-timer Virtual Timer CompareValue Register (EL2)
CNTHVS_TVAL_EL2	3	4	C14	C4	0	See individual bit resets.	64-bit	Counter-timer Secure Virtual Timer TimerValue Register (EL2)
CNTHVS_CTL_EL2	3	4	C14	C4	1	See individual bit resets.	64-bit	Counter-timer Secure Virtual Timer Control Register (EL2)
CNTHVS_CVAL_EL2	3	4	C14	C4	2	See individual bit resets.	64-bit	Counter-timer Secure Virtual Timer CompareValue Register (EL2)
CNTHPS_TVAL_EL2	3	4	C14	C5	0	See individual bit resets.	64-bit	Counter-timer Secure Physical Timer TimerValue Register (EL2)
CNTHPS_CTL_EL2	3	4	C14	C5	1	See individual bit resets.	64-bit	Counter-timer Secure Physical Timer Control Register (EL2)
CNTHPS_CVAL_EL2	3	4	C14	C5	2	See individual bit resets.	64-bit	Counter-timer Secure Physical Timer CompareValue Register (EL2)
CNTPS_TVAL_EL1	3	7	C14	C2	0	See individual bit resets.	64-bit	Counter-timer Physical Secure Timer TimerValue Register
CNTPS_CTL_EL1	3	7	C14	C2	1	See individual bit resets.	64-bit	Counter-timer Physical Secure Timer Control Register
CNTPS_CVAL_EL1	3	7	C14	C2	2	See individual bit resets.	64-bit	Counter-timer Physical Secure Timer CompareValue Register

A.16 AArch64 Trace registers summary

The following summary table provides an overview of all Trace registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-371: Trace registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCTRACEIDR	2	1	C0	C0	1	See individual bit resets.	64-bit	Trace ID Register
TRCVICTLR	2	1	C0	C0	2	See individual bit resets.	64-bit	Trace ViewInst Main Control Register
TRCSEQEVR0	2	1	C0	C0	4	See individual bit resets.	64-bit	Trace Sequencer State Transition Control Register <n>
TRCCNTRLDVR0	2	1	C0	C0	5	See individual bit resets.	64-bit	Trace Counter Reload Value Register <n>
TRCIDR8	2	1	C0	C0	6	0x0000000000000000	64-bit	Trace ID Register 8
TRCIMSPEC0	2	1	C0	C0	7	See individual bit resets.	64-bit	Trace IMP DEF Register 0
TRCPRGCTLR	2	1	C0	C1	0	See individual bit resets.	64-bit	Trace Programming Control Register
TRCVIIECTLR	2	1	C0	C1	2	See individual bit resets.	64-bit	Trace ViewInst Include/Exclude Control Register
TRCSEQEVR1	2	1	C0	C1	4	See individual bit resets.	64-bit	Trace Sequencer State Transition Control Register <n>
TRCCNTRLDVR1	2	1	C0	C1	5	See individual bit resets.	64-bit	Trace Counter Reload Value Register <n>
TRCIDR9	2	1	C0	C1	6	0x0000000000000000	64-bit	Trace ID Register 9
TRCVISSCTLR	2	1	C0	C2	2	See individual bit resets.	64-bit	Trace ViewInst Start/Stop Control Register
TRCSEQEVR2	2	1	C0	C2	4	See individual bit resets.	64-bit	Trace Sequencer State Transition Control Register <n>
TRCIDR10	2	1	C0	C2	6	0x0000000000000000	64-bit	Trace ID Register 10
TRCSTATR	2	1	C0	C3	0	See individual bit resets.	64-bit	Trace Status Register
TRCIDR11	2	1	C0	C3	6	0x0000000000000000	64-bit	Trace ID Register 11
TRCCONFIGR	2	1	C0	C4	0	See individual bit resets.	64-bit	Trace Configuration Register
TRCCNTCTLR0	2	1	C0	C4	5	See individual bit resets.	64-bit	Trace Counter Control Register <n>
TRCIDR12	2	1	C0	C4	6	0x0000000000000000	64-bit	Trace ID Register 12
TRCCNTCTLR1	2	1	C0	C5	5	See individual bit resets.	64-bit	Trace Counter Control Register <n>
TRCIDR13	2	1	C0	C5	6	0x0000000000000000	64-bit	Trace ID Register 13
TRCAUXCTLR	2	1	C0	C6	0	See individual bit resets.	64-bit	Trace Auxiliary Control Register
TRCSEQRSTEVR	2	1	C0	C6	4	See individual bit resets.	64-bit	Trace Sequencer Reset Control Register
TRCSEQSTR	2	1	C0	C7	4	See individual bit resets.	64-bit	Trace Sequencer State Register
TRCEVENTCTL0R	2	1	C0	C8	0	See individual bit resets.	64-bit	Trace Event Control 0 Register
TRCEXTINSELR0	2	1	C0	C8	4	See individual bit resets.	64-bit	Trace External Input Select Register <n>

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCCNTVR0	2	1	C0	C8	5	See individual bit resets.	64-bit	Trace Counter Value Register <n>
TRCIDR0	2	1	C0	C8	7	0x0000000028800EA1	64-bit	Trace ID Register 0
TRCEVENTCTL1R	2	1	C0	C9	0	See individual bit resets.	64-bit	Trace Event Control 1 Register
TRCEXTINSELR1	2	1	C0	C9	4	See individual bit resets.	64-bit	Trace External Input Select Register <n>
TRCCNTVR1	2	1	C0	C9	5	See individual bit resets.	64-bit	Trace Counter Value Register <n>
TRCIDR1	2	1	C0	C9	7	0x000000004100FFF0	64-bit	Trace ID Register 1
TRCRSR	2	1	C0	C10	0	See individual bit resets.	64-bit	Trace Resources Status Register
TRCEXTINSELR2	2	1	C0	C10	4	See individual bit resets.	64-bit	Trace External Input Select Register <n>
TRCIDR2	2	1	C0	C10	7	0x00000000C0001088	64-bit	Trace ID Register 2
TRCEXTINSELR3	2	1	C0	C11	4	See individual bit resets.	64-bit	Trace External Input Select Register <n>
TRCIDR3	2	1	C0	C11	7	0x00000000017F0004	64-bit	Trace ID Register 3
TRCTSCTLR	2	1	C0	C12	0	See individual bit resets.	64-bit	Trace Timestamp Control Register
TRCIDR4	2	1	C0	C12	7	0x0000000011170004	64-bit	Trace ID Register 4
TRCSYNCPR	2	1	C0	C13	0	See individual bit resets.	64-bit	Trace Synchronization Period Register
TRCIDR5	2	1	C0	C13	7	0x00000000284709FF	64-bit	Trace ID Register 5
TRCCCCTLR	2	1	C0	C14	0	See individual bit resets.	64-bit	Trace Cycle Count Control Register
TRCIDR6	2	1	C0	C14	7	0x0000000000000000	64-bit	Trace ID Register 6
TRCBBCTLR	2	1	C0	C15	0	See individual bit resets.	64-bit	Trace Branch Broadcast Control Register
TRCIDR7	2	1	C0	C15	7	0x0000000000000000	64-bit	Trace ID Register 7
TRCSSCCR0	2	1	C1	C0	2	See individual bit resets.	64-bit	Trace Single-shot Comparator Control Register <n>
TRCOSLSR	2	1	C1	C1	4	See individual bit resets.	64-bit	Trace OS Lock Status Register
TRCRSCTLR2	2	1	C1	C2	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR3	2	1	C1	C3	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR4	2	1	C1	C4	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR5	2	1	C1	C5	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR6	2	1	C1	C6	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR7	2	1	C1	C7	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR8	2	1	C1	C8	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCSSCSR0	2	1	C1	C8	2	See individual bit resets.	64-bit	Trace Single-shot Comparator Control Status Register <n>
TRCRSCTLR9	2	1	C1	C9	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR10	2	1	C1	C10	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR11	2	1	C1	C11	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR12	2	1	C1	C12	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR13	2	1	C1	C13	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR14	2	1	C1	C14	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCRSCTLR15	2	1	C1	C15	0	See individual bit resets.	64-bit	Trace Resource Selection Control Register <n>
TRCACVR0	2	1	C2	C0	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR0	2	1	C2	C0	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR1	2	1	C2	C2	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR1	2	1	C2	C2	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCACVR2	2	1	C2	C4	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR2	2	1	C2	C4	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR3	2	1	C2	C6	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR3	2	1	C2	C6	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR4	2	1	C2	C8	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR4	2	1	C2	C8	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR5	2	1	C2	C10	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR5	2	1	C2	C10	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR6	2	1	C2	C12	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR6	2	1	C2	C12	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCACVR7	2	1	C2	C14	0	See individual bit resets.	64-bit	Trace Address Comparator Value Register <n>
TRCACATR7	2	1	C2	C14	2	See individual bit resets.	64-bit	Trace Address Comparator Access Type Register <n>
TRCCIDCVR0	2	1	C3	C0	0	See individual bit resets.	64-bit	Trace Context Identifier Comparator Value Registers <n>
TRCVMIDCVR0	2	1	C3	C0	1	See individual bit resets.	64-bit	Trace Virtual Context Identifier Comparator Value Register <n>
TRCCIDCCTLR0	2	1	C3	C0	2	See individual bit resets.	64-bit	Trace Context Identifier Comparator Control Register 0
TRCVMIDCCTLR0	2	1	C3	C2	2	See individual bit resets.	64-bit	Trace Virtual Context Identifier Comparator Control Register 0
TRCDEVID	2	1	C7	C2	7	`0x0000000000000000`	64-bit	Trace Device Configuration Register
TRCCLAIMSET	2	1	C7	C8	6	See individual bit resets.	64-bit	Trace Claim Tag Set Register
TRCCLAIMCLR	2	1	C7	C9	6	See individual bit resets.	64-bit	Trace Claim Tag Clear Register
TRCAUTHSTATUS	2	1	C7	C14	6	See individual bit resets.	64-bit	Trace Authentication Status Register
TRCDEVARCH	2	1	C7	C15	6	`0x0000000047715A13`	64-bit	Trace Device Architecture Register

A.16.1 TRCIDR8, Trace ID Register 8

Returns the maximum speculation depth of the instruction trace element stream.

Configurations

AArch64 register TRCIDR8 bits [31:0] are architecturally mapped to External register B.5.2 TRCIDR8, Trace ID Register 8 on page 712 bits [31:0].

Attributes

Width

64

Functional group

Trace

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
```

Bit descriptions

Figure A-134: AARCH64_TRCIDR8 bit assignments

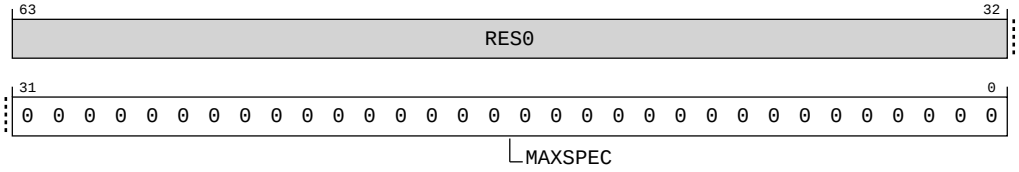


Table A-372: TRCIDR8 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	MAXSPEC	Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of P0 elements in the trace element stream that can be speculative at any time. **0x00000000**	0x00000000

Access

MRS <Xt>, TRCIDR8

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b110

Accessibility

MRS <Xt>, TRCIDR8

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR8;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR8;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR8;
```

A.16.2 TRCIMSPEC0, Trace IMP DEF Register 0

TRCIMSPEC0 shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

Configurations

AArch64 register TRCIMSPEC0 bits [31:0] are architecturally mapped to External register B.5.8 TRCIMSPEC0, Trace IMP DEF Register 0 on page 718 bits [31:0].

Attributes

Width

64

Functional group

Trace

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 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 A-135: AARCH64_TRCIMSPEC0 bit assignments

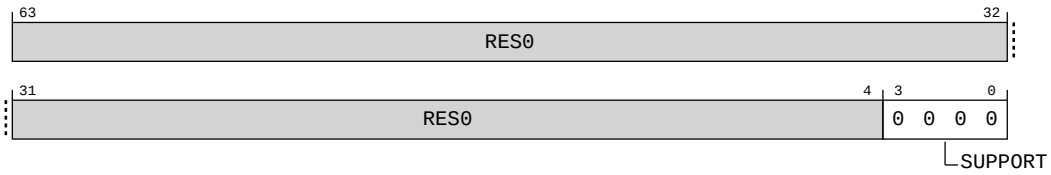


Table A-374: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[63:4]	**RES0**	Reserved	**RES0**
[3:0]	SUPPORT	Indicates whether the implementation supports **IMPLEMENTATION DEFINED** features. **0b0000** No **IMPLEMENTATION DEFINED** features are supported.	0b0000

Access

MRS <Xt>, TRCIMSPEC0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b111

MSR TRCIMSPEC0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b111

Accessibility

MRS <Xt>, TRCIMSPEC0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCIMSPECn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIMSPEC0;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIMSPEC0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIMSPEC0;
```

```
MSR TRCIMSPEC0, <Xt>
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRCIMSPECn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t, 64];
```

A.16.3 TRCIDR10, Trace ID Register 10

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR10 bits [31:0] are architecturally mapped to External register B.5.4 TRCIDR10, Trace ID Register 10 on page 714 bits [31:0].

Attributes

Width

64

Functional group

Trace

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
```

Bit descriptions

Figure A-136: AARCH64_TRCIDR10 bit assignments

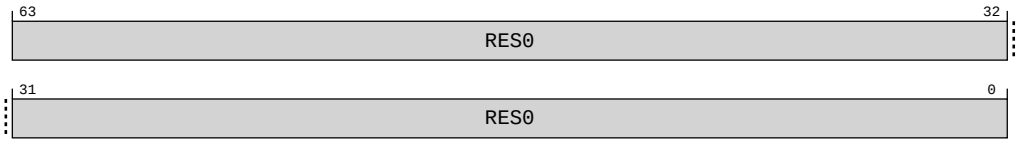


Table A-377: TRCIDR10 bit descriptions

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

Access

MRS <Xt>, TRCIDR10

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b110

Accessibility

MRS <Xt>, TRCIDR10

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR10;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR10;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR10;
```

A.16.4 TRCIDR11, Trace ID Register 11

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR11 bits [31:0] are architecturally mapped to External register B.5.5 TRCIDR11, Trace ID Register 11 on page 715 bits [31:0].

Attributes

Width

64

Functional group

Trace

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
```

Bit descriptions

Figure A-137: AARCH64_TRCIDR11 bit assignments

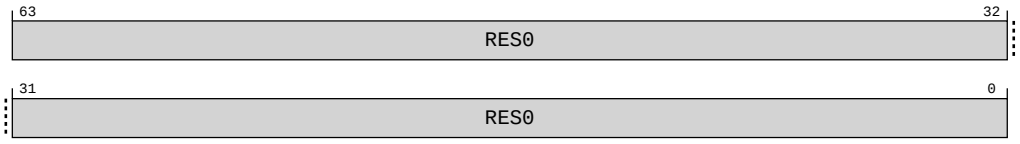


Table A-379: TRCIDR11 bit descriptions

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

Access

MRS <Xt>, TRCIDR11

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0011	0b110

Accessibility

MRS <Xt>, TRCIDR11

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR11;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR11;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR11;
```

A.16.5 TRCIDR12, Trace ID Register 12

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR12 bits [31:0] are architecturally mapped to External register B.5.6 TRCIDR12, Trace ID Register 12 on page 716 bits [31:0].

Attributes

Width

64

Functional group

Trace

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
```

Bit descriptions

Figure A-138: AARCH64_TRCIDR12 bit assignments

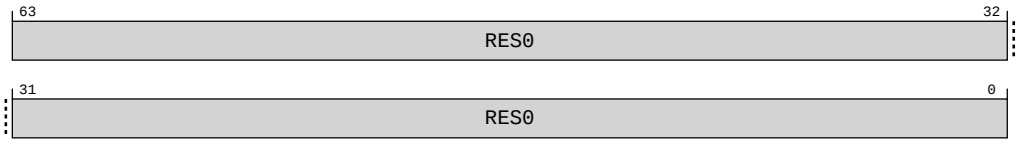


Table A-381: TRCIDR12 bit descriptions

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

Access

MRS <Xt>, TRCIDR12

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0100	0b110

Accessibility

MRS <Xt>, TRCIDR12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR12;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR12;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR12;
```

A.16.6 TRCIDR13, Trace ID Register 13

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR13 bits [31:0] are architecturally mapped to External register B.5.7 TRCIDR13, Trace ID Register 13 on page 717 bits [31:0].

Attributes

Width

64

Functional group

Trace

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
```

Bit descriptions

Figure A-139: AARCH64_TRCIDR13 bit assignments

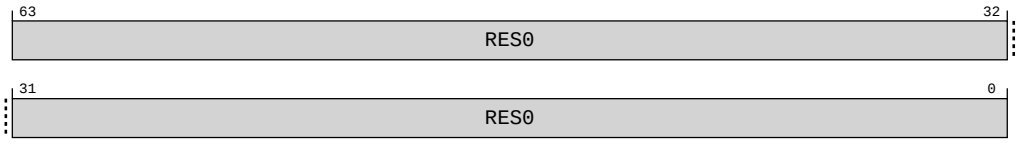


Table A-383: TRCIDR13 bit descriptions

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

Access

MRS <Xt>, TRCIDR13

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0101	0b110

Accessibility

MRS <Xt>, TRCIDR13

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR13;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR13;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR13;
```

A.16.7 TRCIDR0, Trace ID Register 0

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR0 bits [31:0] are architecturally mapped to External register B.5.9 TRCIDR0, Trace ID Register 0 on page 720 bits [31:0].

Attributes

Width

64

Functional group

Trace

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0010 1000 1000 0000 0000 1110 1010 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-140: AARCH64_TRCIDR0 bit assignments

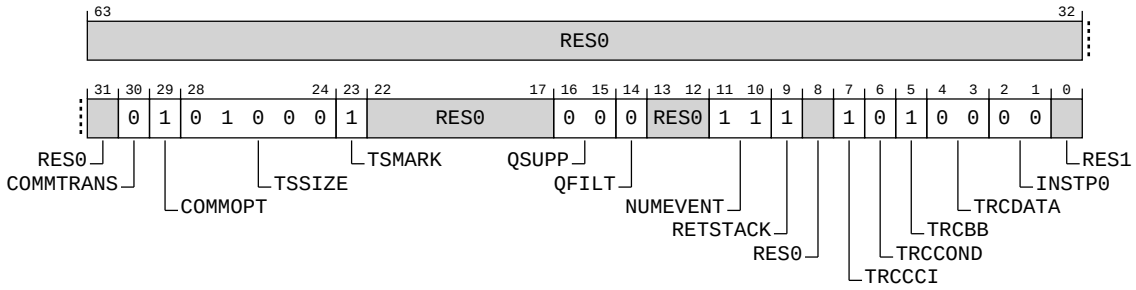


Table A-385: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30]	COMMTRANS	Transaction Start element behavior. **0b0** Transaction Start elements are P0 elements.	0b0
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **0b1** Commit mode 1. Access to this field is: **RAO/WI**	0b1
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. **0b01000** Global timestamping implemented with a 64-bit timestamp value.	0b01000
[23]	TSMARK	Indicates whether Timestamp Marker elements are generated. **0b1** Timestamp Marker elements are generated.	0b1
[22:17]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **0b00** Q element support is not implemented.	0b00
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	0b0
[13:12]	**RES0**	Reserved	**RES0**
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **0b11** The trace unit supports 4 ETEEvents.	0b11
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **0b1** Return stack implemented.	0b1
[8]	**RES0**	Reserved	**RES0**
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **0b1** Cycle counting implemented.	0b1
[6]	TRCCOND	Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b0** Conditional instruction tracing not implemented.	0b0
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. **0b1** Branch broadcasting implemented.	0b1
[4:3]	TRCDATA	Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Tracing of data addresses and data values is not implemented.	0b00
[2:1]	INSTP0	Indicates if load and store instructions are P0 instructions. Load and store instructions as P0 instructions is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Load and store instructions are not P0 instructions.	0b00
[0]	**RES1**	Reserved	**RES1**

Access

```
MRS <Xt>, TRCIDR0
```

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b111

Accessibility

`MRS <Xt>, TRCIDR0`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR0;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR0;
```

A.16.8 TRCIDR1, Trace ID Register 1

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR1 bits [31:0] are architecturally mapped to External register B.5.10 TRCIDR1, Trace ID Register 1 on page 722 bits [31:0].

Attributes

Width

64

Functional group

Trace

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0100 0001 0000 0000 1111 1111 1111 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-141: AARCH64_TRCIDR1 bit assignments

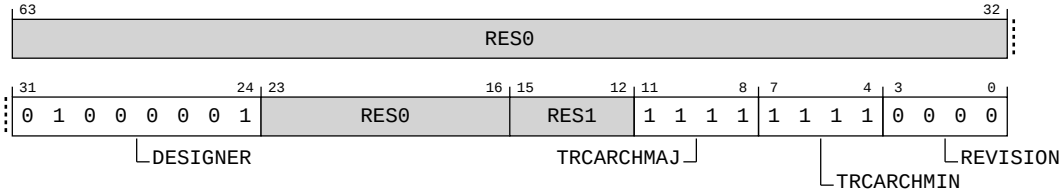


Table A-387: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	DESIGNER	Indicates which company designed the trace unit. The permitted values of this field are the same as MIDR_EL1.Implementer. **0x41** Arm Limited	0x41
[23:16]	**RES0**	Reserved	**RES0**
[15:12]	**RES1**	Reserved	**RES1**
[11:8]	TRCARCHMAJ	Major architecture version. **0b1111** If both TRCIDR1.TRCARCHMAJ and TRCIDR1.TRCARCHMIN == 0xF then refer to TRCDEVARCH.	0b1111
[7:4]	TRCARCHMIN	Minor architecture version. **0b1111** If both TRCIDR1.TRCARCHMAJ and TRCIDR1.TRCARCHMIN == 0xF then refer to TRCDEVARCH.	0b1111
[3:0]	REVISION	Implementation revision. Returns an **IMPLEMENTATION DEFINED** value that identifies the revision of the trace unit. Arm deprecates any use of this field and recommends that implementations set this field to zero. **0b0000** Revision 0	0b0000

Access

```
MRS <Xt>, TRCIDR1
```

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b111

Accessibility

`MRS <Xt>, TRCIDR1`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR1;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR1;
```

A.16.9 TRCIDR2, Trace ID Register 2

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR2 bits [31:0] are architecturally mapped to External register B.5.11 TRCIDR2, Trace ID Register 2 on page 723 bits [31:0].

Attributes

Width

64

Functional group

Trace

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 1100 0000 0000 0000 0001 0000 1000 1000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-142: AARCH64_TRCIDR2 bit assignments

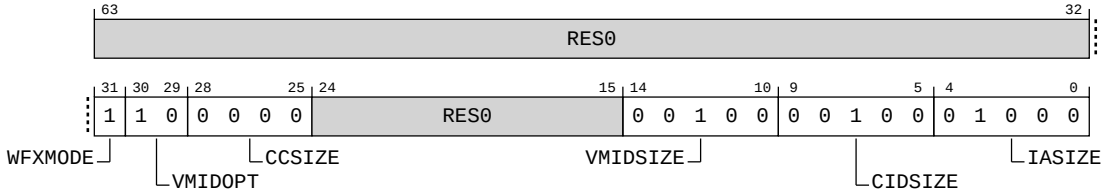


Table A-389: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	WFXMODE	Indicates whether `WFI`, `WFIT`, `WFE`, and `WFET` instructions are classified as P0 instructions: **0b1** `WFI`, `WFIT`, `WFE`, and `WFET` instructions are classified as P0 instructions.	0b1
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. **0b10** Virtual context identifier selection not supported. TRCCONFIGR.VMIDOPT is **RES1**.	0b10
[28:25]	CCSIZE	Indicates the size of the cycle counter. **0b0000** The cycle counter is 12 bits in length.	0b0000
[24:15]	**RES0**	Reserved	**RES0**
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. **0b00100** 32-bit Virtual context identifier size.	0b00100
[9:5]	CIDSIZE	Indicates the Context identifier size. **0b00100** 32-bit Context identifier size.	0b00100
[4:0]	IASIZE	Virtual instruction address size. **0b01000** Maximum of 64-bit instruction address size.	0b01000

Access

`MRS <Xt>, TRCIDR2`

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1010	0b111

Accessibility

`MRS <Xt>, TRCIDR2`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR2;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR2;
```

A.16.10 TRCIDR3, Trace ID Register 3

Returns the base architecture of the trace unit.

Configurations

AArch64 register TRCIDR3 bits [31:0] are architecturally mapped to External register B.5.12 TRCIDR3, Trace ID Register 3 on page 725 bits [31:0].

Attributes

Width

64

Functional group

Trace

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0111 1111 0000 0000 0000 0100
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure A-143: AARCH64_TRCIDR3 bit assignments

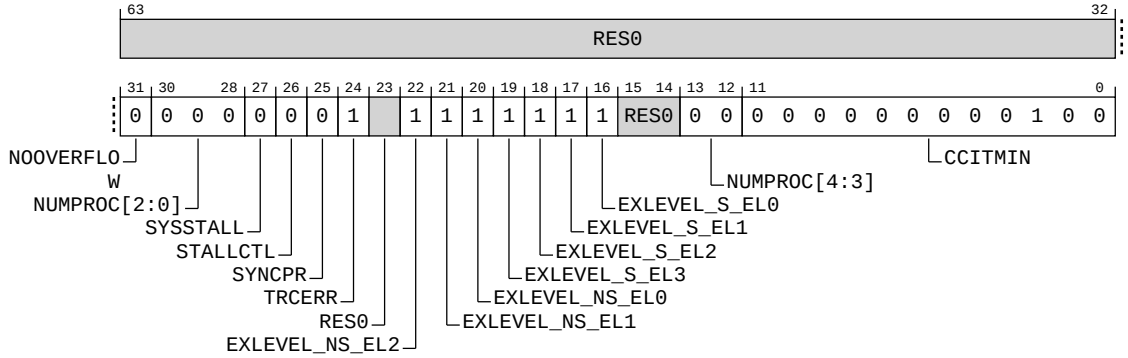


Table A-391: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	0b0
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **0b0** Stalling of the PE is not permitted.	0b0
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **0b0** Stalling of the PE is not implemented.	0b0
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0b0** TRCSYNCPR is read/write so software can change the synchronization period.	0b0
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **0b1** Forced tracing of System Error exceptions is implemented.	0b1
[23]	**RES0**	Reserved	**RES0**
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 is implemented. **0b1** Non-secure EL2 is implemented.	0b1

Bits	Name	Description	Reset
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 is implemented. **0b1** Non-secure EL1 is implemented.	0b1
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 is implemented. **0b1** Non-secure EL0 is implemented.	0b1
[19]	EXLEVEL_S_EL3	Indicates if EL3 is implemented. **0b1** EL3 is implemented.	0b1
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 is implemented. **0b1** Secure EL2 is implemented.	0b1
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 is implemented. **0b1** Secure EL1 is implemented.	0b1
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 is implemented. **0b1** Secure EL0 is implemented.	0b1
[15:14]	**RES0**	Reserved	**RES0**
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. **0b00000** The trace unit can trace one PE.	0b00000
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in TRCCCCTLR.THRESHOLD. **0x004** The minimum value that can be programmed in TRCCCCTLR.THRESHOLD.	0x004

Access

MRS <Xt>, TRCIDR3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1011	0b111

Accessibility

MRS <Xt>, TRCIDR3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR3;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR3;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR3;
```

A.16.11 TRCIDR4, Trace ID Register 4

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR4 bits [31:0] are architecturally mapped to External register B.5.13 TRCIDR4, Trace ID Register 4 on page 727 bits [31:0].

Attributes

Width

64

Functional group

Trace

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0001 0001 0001 0111 0000 0000 0000 0100
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure A-144: AARCH64_TRCIDR4 bit assignments

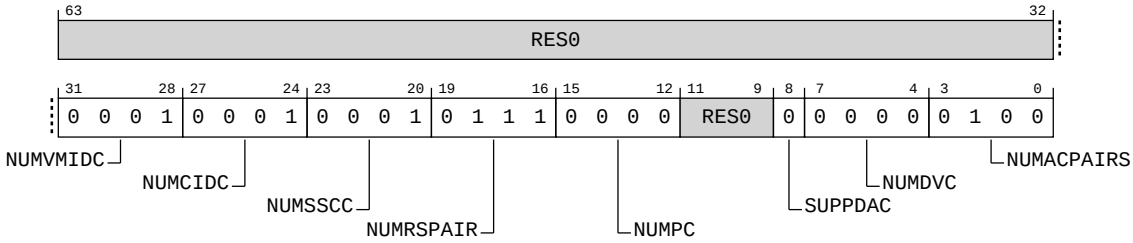


Table A-393: TRCIDR4 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. **0b0001** The number of Virtual Context Identifier Comparators in this implementation.	0b0001
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **0b0001** The number of Context Identifier Comparators in this implementation.	0b0001
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. **0b0001** The number of Single-shot Comparator Controls in this implementation.	0b0001
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0b0111** The number of resource selector pairs in this implementation, minus one.	0b0111
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0b0000** The number of PE Comparator Inputs in this implementation.	0b0000
[11:9]	**RES0**	Reserved	**RES0**
[8]	SUPPDAC	Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0** Data address comparisons not implemented.	0b0
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0000** The number of data value comparators in this implementation.	0b0000
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **0b0100** The number of Address Comparator pairs in this implementation.	0b0100

Access

MRS <Xt>, TRCIDR4

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1100	0b111

Accessibility

MRS <Xt>, TRCIDR4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR4;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR4;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR4;
```

A.16.12 TRCIDR5, Trace ID Register 5

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR5 bits [31:0] are architecturally mapped to External register B.5.14 TRCIDR5, Trace ID Register 5 on page 729 bits [31:0].

Attributes

Width

64

Functional group

Trace

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0010 1000 0100 0111 0000 1001 1111 1111
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure A-145: AARCH64_TRCIDR5 bit assignments

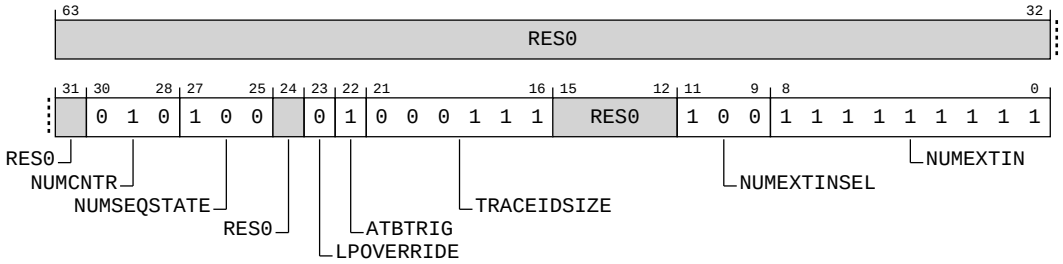


Table A-395: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. **0b010** The number of Counters implemented.	0b010
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **0b100** Four Sequencer states are implemented.	0b100
[24]	**RES0**	Reserved	**RES0**
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **0b0** The trace unit does not support Low-power Override Mode.	0b0
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **0b1** The implementation supports ATB triggers.	0b1
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **0b000111** The implementation supports a 7-bit trace ID.	0b000111
[15:12]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. **0b100** The number of External Input Selector resources implemented.	0b100
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **0b111111111** Unified PMU event selection.	0b111111111

Access

MRS <Xt>, TRCIDR5

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1101	0b111

Accessibility

MRS <Xt>, TRCIDR5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR5;
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR5;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR5;
```

A.16.13 TRCCIDCVR0, Trace Context Identifier Comparator Value Registers <n>

Contains a Context identifier value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

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-146: AARCH64_TRCCIDCVR0 bit assignments

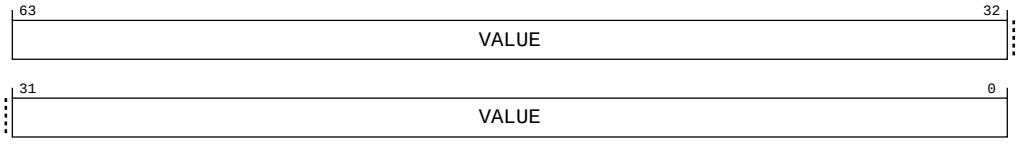


Table A-397: TRCCIDCVR0 bit descriptions

Bits	Name	Description	Reset
[63:0]	VALUE	Context identifier value. The width of this field is indicated by TRCIDR2.CIDSIZE. Unimplemented bits are **RES0**. After a PE Reset, the trace unit assumes that the Context identifier is zero until the PE updates the Context identifier.	`64{x}`

Access

`MRS <Xt>, TRCCIDCVR0`

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0000	0b000

`MSR TRCCIDCVR0, <Xt>`

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0000	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0110 and TRCRSCTLR<a>.CID[n] == 1.
- TRCACATR<a>.CONTEXTTYPE == 0b01 or 0b11 and TRCACATR<a>.CONTEXT == n.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

`MRS <Xt>, TRCCIDCVR0`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCIDCVR[0];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCIDCVR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCIDCVR[0];
```

`MSR TRCCIDCVR0, <Xt>`

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
```

```
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCIDCVR[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if EL3SDDUndefPriority() && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if EL3SDDUndef() then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCIDCVR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCIDCVR[0] = X[t, 64];
```

Appendix B External registers

This appendix contains the descriptions for the C1-Ultra 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 all 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.

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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
0x120	AMEVCNTR14 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
0x128	AMEVCNTR15 [63:0]	0x0000000000000000	64-bit	Activity Monitors Event Counter Registers 1
0x400	AMEVTYPER00	0x00000011	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	0x00004004	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	0x00000008	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	0x00004005	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
0x490	AMEVTYPER14	0x00003200	32-bit	Activity Monitors Event Type Registers 1
0x494	AMEVTYPER15	0x00003202	32-bit	Activity Monitors Event Type Registers 1

Offset	Name	Reset	Width	Description
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	0x00000604	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	0x11003F09	32-bit	Activity Monitors Configuration Register
0xE04	AMCR	See individual bit resets.	32-bit	Activity Monitors Control Register
0xE08	AMIIDR	0xD8C1043B	32-bit	Activity Monitors Implementation Identification Register
0xFA8	AMDEVAFF0	See individual bit resets.	32-bit	Activity Monitors Device Affinity Register 0
0xFAC	AMDEVAFF1	See individual bit resets.	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	0x0000008C	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	0x00000000	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 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.

Attributes

Width

64

Component

AMU

Register offset

0x118

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
```

Bit descriptions

Figure B-1: AMU_AMEVCNTR13 bit assignments

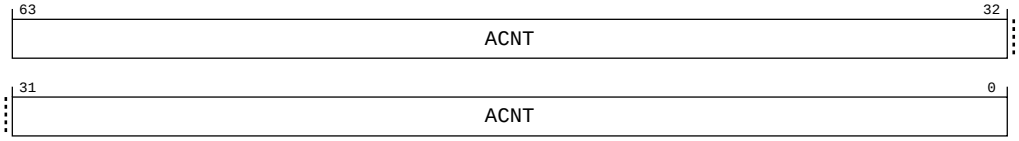


Table B-2: AMEVCNTR13 bit descriptions

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

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:

RO

B.1.2 AMEVCNTR14, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counter 4.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offset

0x120

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
```

Bit descriptions

Figure B-2: AMU_AMEVCNTR14 bit assignments

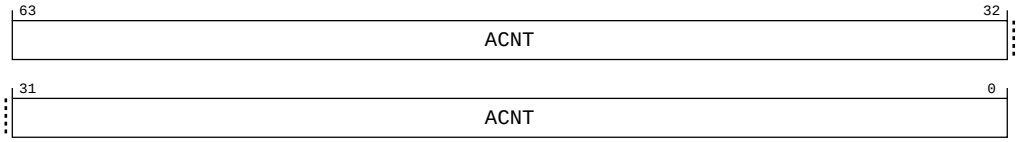


Table B-4: AMEVCNTR14 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of Auxiliary activity monitor event counter 4.	0x0000000000000000

Accessibility

If 4 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR14 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

AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x120	AMEVCNTR14	63:0

This interface is accessible as follows:

RO

B.1.3 AMEVCNTR15, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counter 5.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offset

0x128

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
```

Bit descriptions

Figure B-3: AMU_AMEVCNTR15 bit assignments

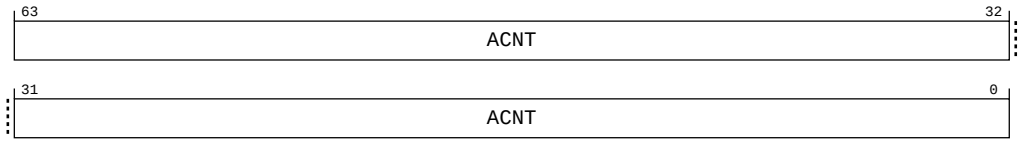


Table B-6: AMEVCNTR15 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of Auxiliary activity monitor event counter 5.	0x0000000000000000

Accessibility

If 5 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR15 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

AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x128	AMEVCNTR15	63:0

This interface is accessible as follows:

RO

B.1.4 AMEVTYPER00, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMU.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 0000 0000 0001 0001
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-4: AMU_AMEVTYPER00 bit assignments

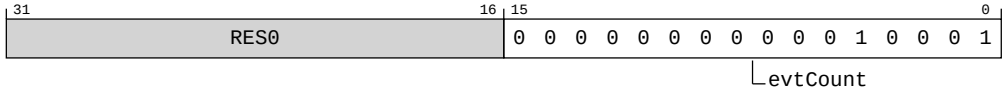


Table B-8: AMEVTYPER00 bit descriptions

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.AMEVCNTR00. The value of this field is architecturally mandated for each architected counter. **0x0011** Processor frequency cycles.	0x0011

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

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:

RO

B.1.5 AMEVTYPER01, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMU.AMEVCNTR01 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

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-5: AMU_AMEVTYPER01 bit assignments

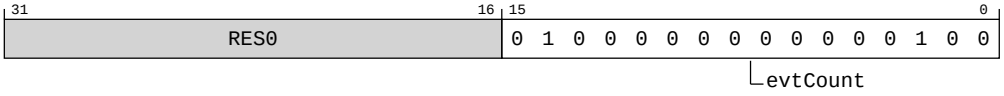


Table B-10: AMEVTYPER01 bit descriptions

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. **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

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:

RO

B.1.6 AMEVTYPER02, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMU.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 0000 0000 0000 1000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-6: AMU_AMEVTYPER02 bit assignments

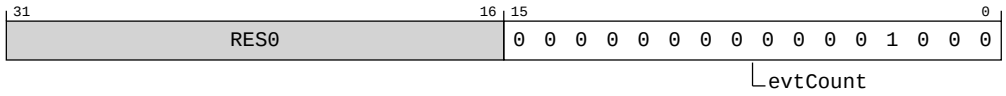


Table B-12: AMEVTYPER02 bit descriptions

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.AMEVCNTR02. The value of this field is architecturally mandated for each architected counter. **0x0008** Instructions retired.	0x0008

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

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:

RO

B.1.7 AMEVTYPER03, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AMU.AMEVCNTR03 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x40C

Access type

RO

Reset value

```
0000 0000 0000 0000 0100 0000 0000 0101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-7: AMU_AMEVTYPER03 bit assignments

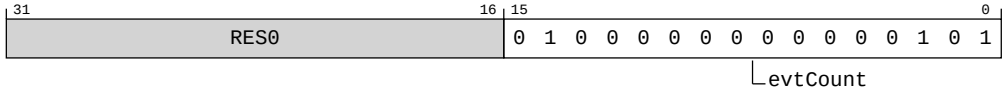


Table B-14: AMEVTYPER03 bit descriptions

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. **0x4005** Memory stall cycles.	0x4005

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

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:

RO

B.1.8 AMEVTYPER10, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMU.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-8: AMU_AMEVTYPER10 bit assignments

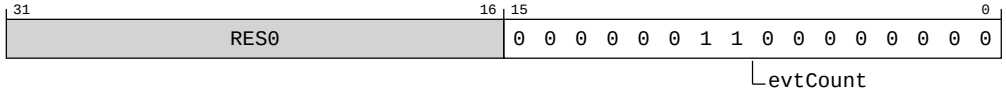


Table B-16: AMEVTYPER10 bit descriptions

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. **0x0300** Gear 0 (MPMM bank 0) period threshold exceeded	0x0300

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

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:

RO

B.1.9 AMEVTYPER11, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMU.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
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-9: AMU_AMEVTYPER11 bit assignments

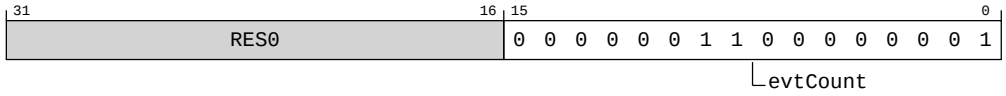


Table B-18: AMEVTYPER11 bit descriptions

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. **0x0301** Gear 1 (MPMM bank 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

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:

RO

B.1.10 AMEVTYPER12, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMU.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-10: AMU_AMEVTYPER12 bit assignments

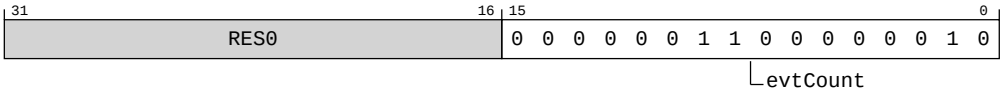


Table B-20: AMEVTYPER12 bit descriptions

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. **0x0302** Gear 2 (MPMM bank 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*.


Note

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:

RO

B.1.11 AMEVTYPER13, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMU.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
```

Bit descriptions

Figure B-11: AMU_AMEVTYPER13 bit assignments

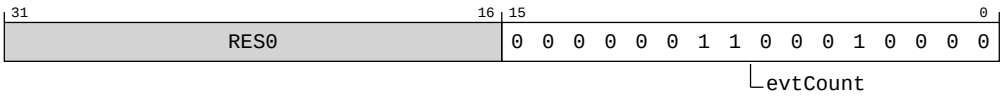


Table B-22: AMEVTYPER13 bit descriptions

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. **0x0310** CPU Activity Count	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

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:

RO

B.1.12 AMEVTYPER14, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMU.AMEVCNTR14 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x490

Access type

RO

Reset value

```
0000 0000 0000 0000 0011 0010 0000 0000
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-12: AMU_AMEVTYPER14 bit assignments

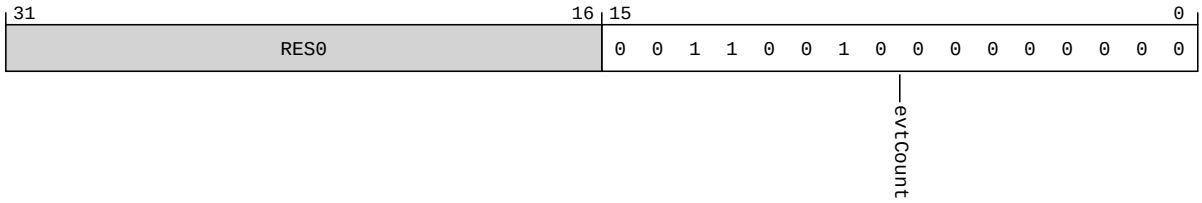


Table B-24: AMEVTYPER14 bit descriptions

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.AMEVCNTR14. **0x3200** CPU is stalling because of SME2 unit backpressure, for any reason	0x3200

Accessibility

If 4 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER14 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

AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x490	AMEVTYPER14	None

This interface is accessible as follows:

RO

B.1.13 AMEVTYPER15, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AMU.AMEVCNTR15 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x494

Access type

RO

Reset value

```
0000 0000 0000 0000 0011 0010 0000 0010
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-13: AMU_AMEVTYPER15 bit assignments

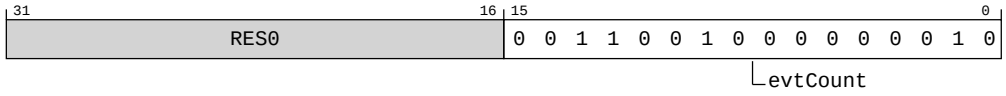


Table B-26: AMEVTYPER15 bit descriptions

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.AMEVCNTR15. **0x3202** CPU is stalling because of SME2 unit backpressure, waiting for arbitration	0x3202

Accessibility

If 5 is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER15 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

AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x494	AMEVTYPER15	None

This interface is accessible as follows:

RO

B.1.14 AMCGCR, Activity Monitors Counter Group Configuration Register

Provides information on the number of activity monitor event counters implemented within each counter group.

Configurations

External register AMCGCR bits [31:0] are architecturally mapped to AArch64 System register A.1.2 AMCGCR_EL0, Activity Monitors Counter Group Configuration Register on page 205 bits [31:0].

Attributes

Width

32

Component

AMU

Register offset

0xCE0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0110 0000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-14: AMU_AMCGCR bit assignments

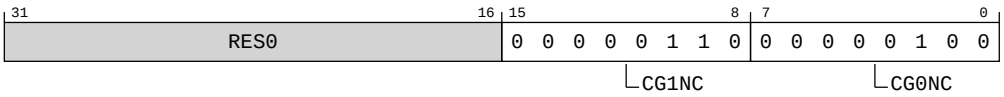


Table B-28: AMCGCR bit descriptions

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. **0x06** Six counters in the auxiliary counter group	0x06
[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:

RO

B.1.15 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

External register AMCFGR bits [31:0] are architecturally mapped to AArch64 System register A.1.1 AMCFGR_EL0, Activity Monitors Configuration Register on page 203 bits [31:0].

Attributes

Width

32

Component

AMU

Register offset

0xE00

Access type

RO

Reset value

```
0001 0001 0000 0000 0011 1111 0000 1001
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-15: AMU_AMCFGR bit assignments

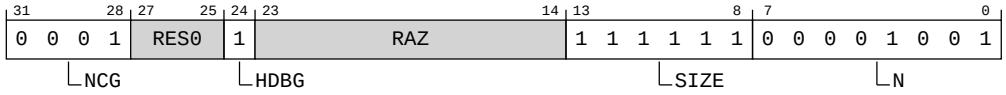


Table B-30: AMCFGR bit descriptions

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** 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. **0x09** Ten activity monitor event counters	0x09

Accessibility

Component	Offset	Instance	Range
AMU	0xE00	AMCFGR	None

This interface is accessible as follows:

RO

B.1.16 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 1100 0001 0000 0100 0011 1011
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-16: AMU_AMIIDR bit assignments

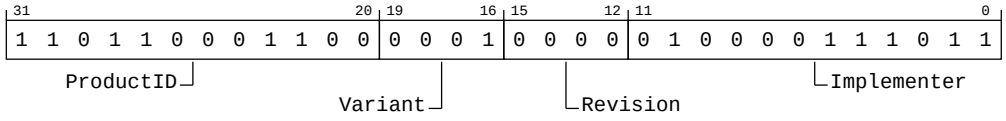


Table B-32: AMIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	This field is an AMU part identifier. **0xD8C** C1-Ultra	0xD8C
[19:16]	Variant	This field distinguishes product variants or major revisions of the product. **0b0001** r1p0	0b0001
[15:12]	Revision	This field distinguishes minor revisions of the product. **0b0000** r1p0	0b0000

Bits	Name	Description	Reset
[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:

RO

B.1.17 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-17: AMU_AMDEVARCH bit assignments

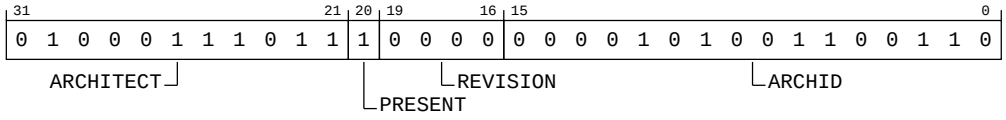


Table B-34: AMDEVARCH bit descriptions

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`
[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.	`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:

RO

B.1.18 AMDEVTYPE, Activity Monitors Device Type Register

Indicates to a debugger that this component is part of a PE's performance monitor interface.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFCC

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-18: AMU_AMDEVTYPE bit assignments

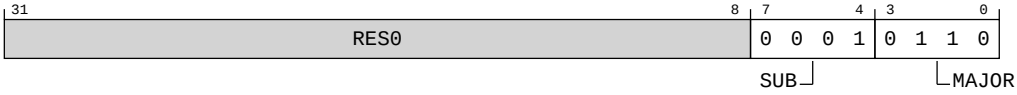


Table B-36: AMDEVTYPE bit descriptions

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:

RO

B.1.19 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-19: AMU_AMPIDR4 bit assignments

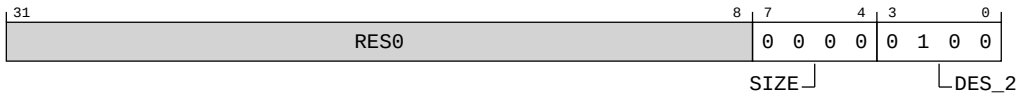


Table B-38: AMPIDR4 bit descriptions

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

Accessibility

Component	Offset	Instance	Range
AMU	0xFD0	AMPIDR4	None

This interface is accessible as follows:

RO

B.1.20 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 1100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-20: AMU_AMPIDR0 bit assignments

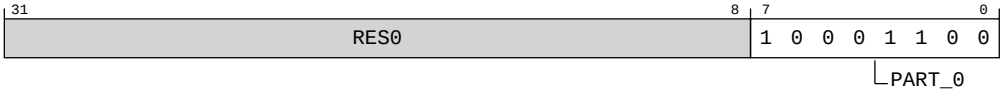


Table B-40: AMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0x8C** C1-Ultra	0x8C

Accessibility

Component	Offset	Instance	Range
AMU	0xFE0	AMPIDR0	None

This interface is accessible as follows:

RO

B.1.21 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

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-21: AMU_AMPIDR1 bit assignments

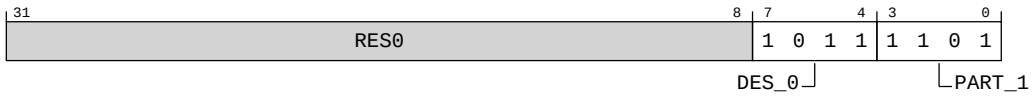


Table B-42: AMPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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-Ultra	`0b1101`

Accessibility

Component	Offset	Instance	Range
AMU	0xFE4	AMPIDR1	None

This interface is accessible as follows:

RO

B.1.22 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-22: AMU_AMPIDR2 bit assignments

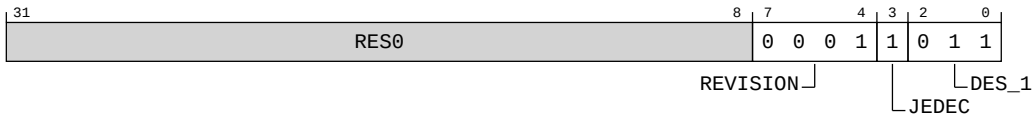


Table B-44: AMPIDR2 bit descriptions

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** r1p0	0b0001
[3]	JEDEC	Indicates a JEP106 identity code is used. **0b1**	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	Instance	Range
AMU	0xFE8	AMPIDR2	None

This interface is accessible as follows:

RO

B.1.23 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 0000 0000
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-23: AMU_AMPIDR3 bit assignments

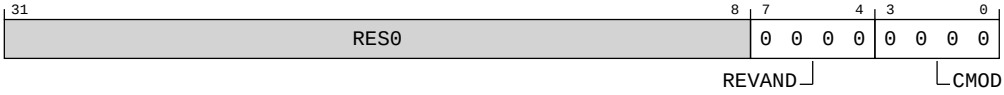


Table B-46: AMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Part minor revision. Parts using AMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0000** r1p0	0b0000
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000** The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Instance	Range
AMU	0xFEC	AMPIDR3	None

This interface is accessible as follows:

RO

B.1.24 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

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-24: AMU_AMCIDR0 bit assignments

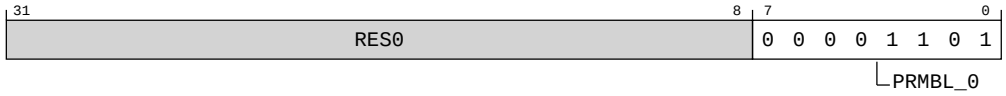


Table B-48: AMCIDR0 bit descriptions

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:

RO

B.1.25 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-25: AMU_AMCIDR1 bit assignments

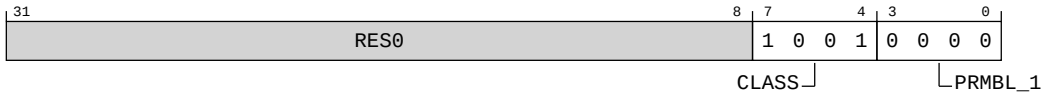


Table B-50: AMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight component.	0b1001
[3:0]	PRMBL_1	Preamble. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
AMU	0xFF4	AMCIDR1	None

This interface is accessible as follows:

RO

B.1.26 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
```

Bit descriptions

Figure B-26: AMU_AMCIDR2 bit assignments

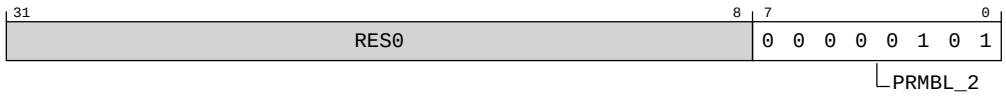


Table B-52: AMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	PRMBL_2	Preamble. **0x05**	0x05

Accessibility

Component	Offset	Instance	Range
AMU	0xFF8	AMCIDR2	None

This interface is accessible as follows:

RO

B.1.27 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

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-27: AMU_AMCIDR3 bit assignments

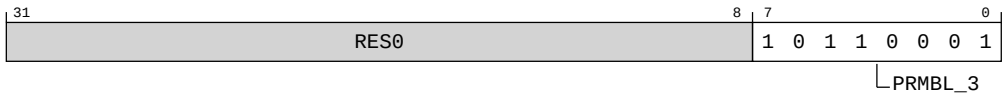


Table B-54: AMCIDR3 bit descriptions

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:

RO

B.2 External CTI registers summary

The following summary table provides an overview of all memory-mapped CTI registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-56: CTI registers summary

Offset	Name	Reset	Width	Description
0x150	CTIDEVCTL	See individual bit resets.	32-bit	CTI Device Control register

B.3 External CoreROM registers summary

The following summary table provides an overview of all memory-mapped CoreROM 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-57: CoreROM registers summary

Offset	Name	Reset	Width	Description
0x000	COREROM_ROMENTRY0	0x00010003	32-bit	Core ROM table Entry 0
0x004	COREROM_ROMENTRY1	0x00020003	32-bit	Core ROM table Entry 1
0x008	COREROM_ROMENTRY2	0x00030003	32-bit	Core ROM table Entry 2
0x00C	COREROM_ROMENTRY3	0x00040003	32-bit	Core ROM table Entry 3
0xFB8	COREROM_AUTHSTATUS	0x00000000	32-bit	Core ROM table Authentication Status Register
0xFBC	COREROM_DEVARCH	0x47700AF7	32-bit	Core ROM table Device Architecture Register
0xFCC	COREROM_DEVTYPE	0x00000000	32-bit	Core ROM table Device Type Register
0xFD0	COREROM_PIDR4	0x00000004	32-bit	Core ROM table Peripheral Identification Register 4
0xFE0	COREROM_PIDR0	0x0000008C	32-bit	Core ROM table Peripheral Identification Register 0
0xFE4	COREROM_PIDR1	0x000000BD	32-bit	Core ROM table Peripheral Identification Register 1
0xFE8	COREROM_PIDR2	0x0000001B	32-bit	Core ROM table Peripheral Identification Register 2
0xFEC	COREROM_PIDR3	0x00000000	32-bit	Core ROM table Peripheral Identification Register 3
0xFF0	COREROM_CIDR0	0x0000000D	32-bit	Core ROM table Component Identification Register 0
0xFF4	COREROM_CIDR1	0x00000090	32-bit	Core ROM table Component Identification Register 1
0xFF8	COREROM_CIDR2	0x00000005	32-bit	Core ROM table Component Identification Register 2
0xFFC	COREROM_CIDR3	0x000000B1	32-bit	Core ROM table Component Identification Register 3

B.3.1 COREROM_ROMENTRY0, Core ROM table Entry 0

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0x000

Access type

RO

Reset value

```
0000 0000 0000 0001 0000 0000 0000 0011
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-28: EXT_COREROM_ROMENTRY0 bit assignments

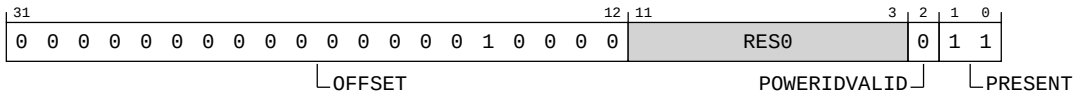


Table B-58: COREROM_ROMENTRY0 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). **0x00010** Core DBG component at address 0x1_0000.	0x00010
[11:3]	**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. **0b11** The ROM Entry is present.	0b11

Accessibility

Component	Offset	Range
CoreROM	0x000	None

This interface is accessible as follows:

RO

B.3.2 COREROM_ROMENTRY1, Core ROM table Entry 1

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0x004

Access type

RO

Reset value

```
0000 0000 0000 0010 0000 0000 0000 0011
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-29: EXT_COREROM_ROMENTRY1 bit assignments

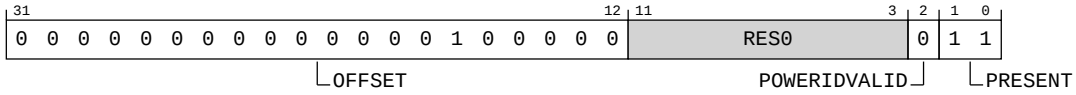


Table B-60: COREROM_ROMENTRY1 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). **0x00020** CORE PMU component at address 0x2_0000.	0x00020
[11:3]	**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. **0b11** The ROM Entry is present.	0b11

Accessibility

Component	Offset	Range
CoreROM	0x004	None

This interface is accessible as follows:

RO

B.3.3 COREROM_ROMENTRY2, Core ROM table Entry 2

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0x008

Access type

RO

Reset value

```
0000 0000 0000 0011 0000 0000 0000 0011
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-30: EXT_COREROM_ROMENTRY2 bit assignments

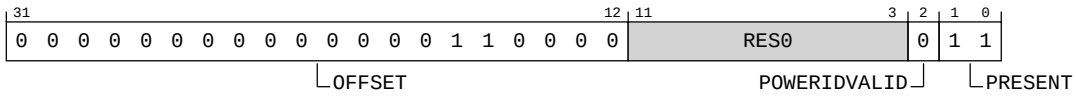


Table B-62: COREROM_ROMENTRY2 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). **0x00030** Core trace unit component at address 0x3_0000.	0x00030
[11:3]	**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. **0b11** The ROM Entry is present.	0b11

Accessibility

Component	Offset	Range
CoreROM	0x008	None

This interface is accessible as follows:

RO

B.3.4 COREROM_ROMENTRY3, Core ROM table Entry 3

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0x00C

Access type

RO

Reset value

```
0000 0000 0000 0100 0000 0000 0000 0011
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-31: EXT_COREROM_ROMENTRY3 bit assignments

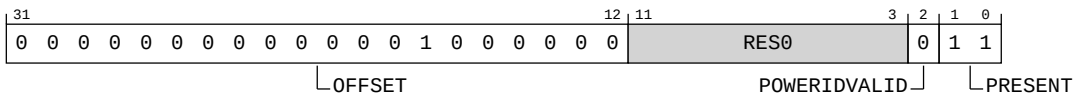


Table B-64: COREROM_ROMENTRY3 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). **0x00040** Core ELA component at address 0x4_0000. When the core is configured without ELA, this field is set to 0x000.	0x00040
[11:3]	**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. **0b11** The ROM Entry is present. When the core is configured without ELA, this field is set to 0x0.	0b11

Accessibility

Component	Offset	Range
CoreROM	0x00C	None

This interface is accessible as follows:

RO

B.3.5 COREROM_AUTHSTATUS, Core ROM table Authentication Status Register

Provides information about the state of the authentication interface for debug.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFB8

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-32: EXT_COREROM_AUTHSTATUS bit assignments

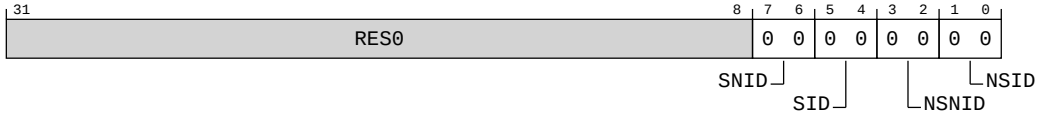


Table B-66: COREROM_AUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:6]	SNID	Secure Non-invasive Debug. **0b00**	0b00
[5:4]	SID	Secure Invasive Debug. **0b00**	0b00
[3:2]	NSNID	Non-secure Non-invasive Debug. **0b00** Debug level is not supported.	0b00
[1:0]	NSID	Non-secure Invasive Debug. **0b00** Debug level is not supported.	0b00

Accessibility

Component	Offset	Range
CoreROM	0xFB8	None

This interface is accessible as follows:

RO

B.3.6 COREROM_DEVARCH, Core ROM table Device Architecture Register

Identifies the architect and architecture of a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFBC

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-33: EXT_COREROM_DEVARCH bit assignments

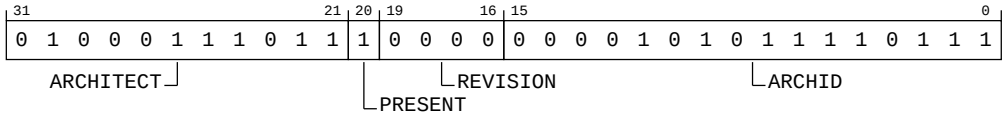


Table B-68: COREROM_DEVARCH bit descriptions

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 COREROM_DEVTYPE and COREROM_DEVID to determine further information about the ROM Table.	0x0AF7

Accessibility

Component	Offset	Range
CoreROM	0xFBC	None

This interface is accessible as follows:

RO

B.3.7 COREROM_DEVTYPE, Core ROM table Device Type Register

A debugger can use DEVTYPE to obtain information about a component that has an unrecognized part number.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFCC

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-34: EXT_COREROM_DEVTYPE bit assignments

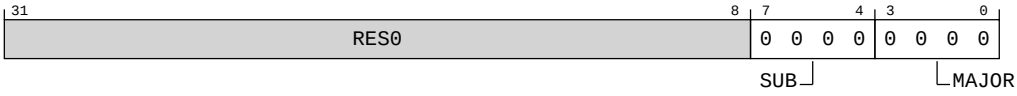


Table B-70: COREROM_DEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Sub number **0b0000** Other, undefined.	0b0000
[3:0]	MAJOR	Major number **0b0000** Miscellaneous.	0b0000

Accessibility

Component	Offset	Range
CoreROM	0xFCC	None

This interface is accessible as follows:

RO

B.3.8 COREROM_PIDR4, Core ROM table Peripheral Identification Register 4

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFD0

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-35: EXT_COREROM_PIDR4 bit assignments

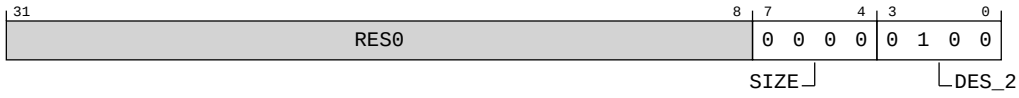


Table B-72: COREROM_PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	SIZE	4KB count. **0b0000** The component uses a single 4KB block.	0b0000
[3:0]	DES_2	JEP106 continuation code. **0b0100** Arm Limited	0b0100

Accessibility

Component	Offset	Range
CoreROM	0xFD0	None

This interface is accessible as follows:

RO

B.3.9 COREROM_PIDR0, Core ROM table Peripheral Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFE0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1000 1100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-36: EXT_COREROM_PIDR0 bit assignments

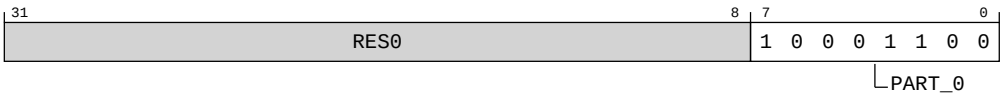


Table B-74: COREROM_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number bits [7:0]. **0x8C** C1-Ultra	0x8C

Accessibility

Component	Offset	Range
CoreROM	0xFE0	None

This interface is accessible as follows:

RO

B.3.10 COREROM_PIDR1, Core ROM table Peripheral Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFE4

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-37: EXT_COREROM_PIDR1 bit assignments

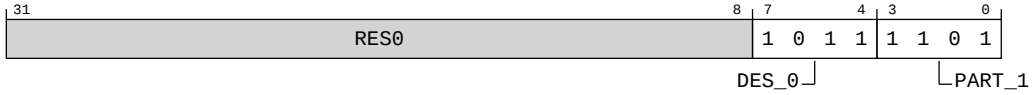


Table B-76: COREROM_PIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	JEP106 identification code bits [3:0]. **0b1011** Arm Limited	0b1011
[3:0]	PART_1	Part number bits [11:8]. **0b1101** C1-Ultra	0b1101

Accessibility

Component	Offset	Range
CoreROM	0xFE4	None

This interface is accessible as follows:

RO

B.3.11 COREROM_PIDR2, Core ROM table Peripheral Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFE8

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-38: EXT_COREROM_PIDR2 bit assignments

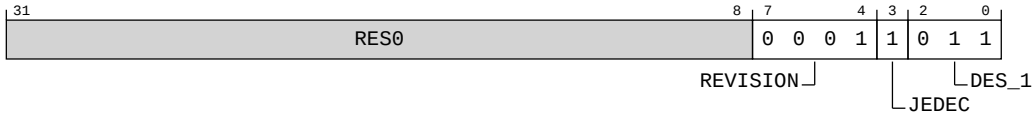


Table B-78: COREROM_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Component revision. **0b0001** r1p0	0b0001
[3]	JEDEC	JEDEC assignee. **0b1** JEDEC-assignee values is used.	0b1
[2:0]	DES_1	JEP106 identification code bits [6:4]. **0b011** Arm Limited	0b011

Accessibility

Component	Offset	Range
CoreROM	0xFE8	None

This interface is accessible as follows:

RO

B.3.12 COREROM_PIDR3, Core ROM table Peripheral Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFEC

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-39: EXT_COREROM_PIDR3 bit assignments

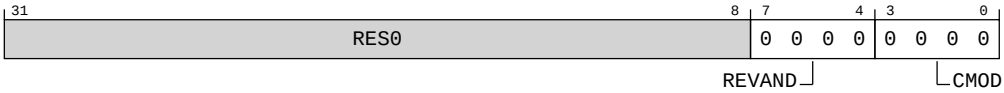


Table B-80: COREROM_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Minor errata fixes. **0b0000** r1p0	0b0000
[3:0]	CMOD	Customer Modified. **0b0000** The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Range
CoreROM	0xFEC	None

This interface is accessible as follows:

RO

B.3.13 COREROM_CIDR0, Core ROM table Component Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFF0

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-40: EXT_COREROM_CIDR0 bit assignments

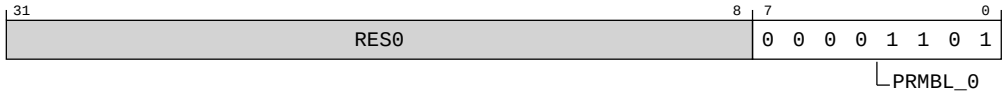


Table B-82: COREROM_CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	CoreSight component identification preamble. **0x0D** CoreSight component identification preamble.	0x0D

Accessibility

Component	Offset	Range
CoreROM	0xFF0	None

This interface is accessible as follows:

RO

B.3.14 COREROM_CIDR1, Core ROM table Component Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFF4

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-41: EXT_COREROM_CIDR1 bit assignments

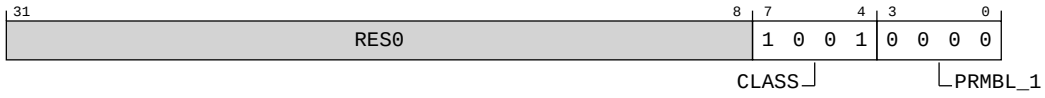


Table B-84: COREROM_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	CLASS	CoreSight component class. **0b1001** CoreSight component.	0b1001
[3:0]	PRMBL_1	CoreSight component identification preamble. **0b0000** CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Range
CoreROM	0xFF4	None

This interface is accessible as follows:

RO

B.3.15 COREROM_CIDR2, Core ROM table Component Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFF8

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-42: EXT_COREROM_CIDR2 bit assignments

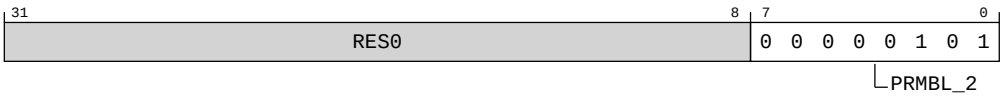


Table B-86: COREROM_CIDR2 bit descriptions

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
CoreROM	0xFF8	None

This interface is accessible as follows:

RO

B.3.16 COREROM_CIDR3, Core ROM table Component Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFFC

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-43: EXT_COREROM_CIDR3 bit assignments

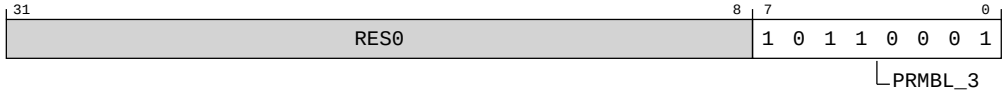


Table B-88: COREROM_CIDR3 bit descriptions

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
CoreROM	0xFFC	None

This interface is accessible as follows:

RO

B.4 External Debug registers summary

The following summary table provides an overview of all memory-mapped Debug registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-90: Debug registers summary

Offset	Name	Reset	Width	Description
0x020	EDESR	See individual bit resets.	32-bit	External Debug Event Status Register
0x024	EDECR	See individual bit resets.	32-bit	External Debug Execution Control Register

Offset	Name	Reset	Width	Description
0x030	EDWAR	See individual bit resets.	64-bit	External Debug Watchpoint Address Register
0x038	EDHSR	See individual bit resets.	64-bit	External Debug Halting Syndrome Register
0x080	DBGDTRRX_EL0	See individual bit resets.	32-bit	Debug Data Transfer Register, Receive
0x084	EDITR	See individual bit resets.	32-bit	External Debug Instruction Transfer Register
0x088	EDSCR	See individual bit resets.	32-bit	External Debug Status and Control Register
0x08C	DBGDTRTX_EL0	See individual bit resets.	32-bit	Debug Data Transfer Register, Transmit
0x090	EDRCR	See individual bit resets.	32-bit	External Debug Reserve Control Register
0x098	EDECCR	See individual bit resets.	32-bit	External Debug Exception Catch Control Register
0x300	OSLAR_EL1	See individual bit resets.	32-bit	OS Lock Access Register
0x310	EDPRCR	See individual bit resets.	32-bit	External Debug Power/Reset Control Register
0x314	EDPRSR	See individual bit resets.	32-bit	External Debug Processor Status Register
0x400	DBGBVR0_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x408	DBGBCR0_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x410	DBGBVR1_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x418	DBGBCR1_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x420	DBGBVR2_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x428	DBGBCR2_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x430	DBGBVR3_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x438	DBGBCR3_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x440	DBGBVR4_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x448	DBGBCR4_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x450	DBGBVR5_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x458	DBGBCR5_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x800	DBGWVR0_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x808	DBGWCR0_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x810	DBGWVR1_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x818	DBGWCR1_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x820	DBGWVR2_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x828	DBGWCR2_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x830	DBGWVR3_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x838	DBGWCR3_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0xD00	MIDR_EL1	`0x411FD8C0`	32-bit	Main ID Register
0xD20	EDPFR	See individual bit resets.	64-bit	External Debug Processor Feature Register
0xD28	EDDFR	See individual bit resets.	64-bit	External Debug Feature Register
0xD48	EDDFR1	See individual bit resets.	64-bit	External Debug Feature Register 1
0xD60	EDAA32PFR	`0x0000000000000000`	64-bit	External Debug Auxiliary Processor Feature Register
0xF00	EDITCTRL	See individual bit resets.	32-bit	External Debug Integration mode Control register
0xFA0	DBGCLAIMSET_EL1	See individual bit resets.	32-bit	Debug CLAIM Tag Set Register
0xFA4	DBGCLAIMCLR_EL1	`0x00000000`	32-bit	Debug CLAIM Tag Clear Register
0xFA8	EDDEVAFF0	See individual bit resets.	32-bit	External Debug Device Affinity register 0

Offset	Name	Reset	Width	Description
0xFAC	EDDEVAFF1	See individual bit resets.	32-bit	External Debug Device Affinity register 1
0xFB0	EDLAR	See individual bit resets.	32-bit	External Debug Lock Access Register
0xFB4	EDLSR	See individual bit resets.	32-bit	External Debug Lock Status Register
0xFB8	DBGAUTHSTATUS_EL1	See individual bit resets.	32-bit	Debug Authentication Status Register
0xFBC	EDDEVARCH	0x4770AA15	32-bit	External Debug Device Architecture Register
0xFC0	EDDEVID2	0x00000000	32-bit	External Debug Device ID register 2
0xFC4	EDDEVID1	0x00000010	32-bit	External Debug Device ID Register 1
0xFC8	EDDEVID	0x00000010	32-bit	External Debug Device ID register 0
0xFCC	EDDEVTYPE	0x00000015	32-bit	External Debug Device Type register
0xFD0	EDPIDR4	0x00000004	32-bit	External Debug Peripheral Identification Register 4
0xFE0	EDPIDR0	0x0000008C	32-bit	External Debug Peripheral Identification Register 0
0xFE4	EDPIDR1	0x000000BD	32-bit	External Debug Peripheral Identification Register 1
0xFE8	EDPIDR2	0x0000001B	32-bit	External Debug Peripheral Identification Register 2
0xFEC	EDPIDR3	0x00000000	32-bit	External Debug Peripheral Identification Register 3
0xFF0	EDCIDR0	0x0000000D	32-bit	External Debug Component Identification Register 0
0xFF4	EDCIDR1	0x00000090	32-bit	External Debug Component Identification Register 1
0xFF8	EDCIDR2	0x00000005	32-bit	External Debug Component Identification Register 2
0xFFC	EDCIDR3	0x000000B1	32-bit	External Debug Component Identification Register 3

B.4.1 EDHSR, External Debug Halting Syndrome Register

Holds syndrome information for a debug event.

Configurations

The value of this register is **UNKNOWN** if the PE is in Non-debug state, or if EDSCR.STATUS is not 0b101011.

Attributes

Width

64

Component

Debug

Register offset

0x038

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 xxxx xxxx x000 0x00 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-44: EXT_EDHSR bit assignments

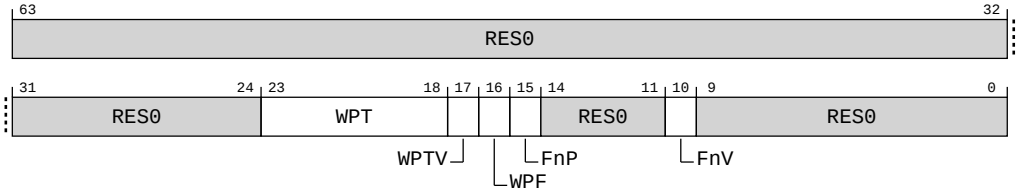


Table B-91: EDHSR bit descriptions

Bits	Name	Description	Reset
[63:24]	**RES0**	Reserved	**RES0**
[23:18]	WPT	Watchpoint number. When EDHSR.WPTV is 1, holds the index of a watchpoint that triggered the Watchpoint debug event.	6{x}
[17]	WPTV	Watchpoint number valid. **0b0** EDHSR.WPT field is not valid, and holds an **UNKNOWN** value. **0b1** EDHSR.WPT field is valid, and holds the number of a watchpoint that triggered the Watchpoint debug event.	x
[16]	WPF	Watchpoint might be false-positive. **0b0** The watchpoint matched an address or address range that was accessed by the instruction. **0b1** The watchpoint matched an address or address range that might not have been accessed by the instruction.	x
[15]	FnP	EDWAR not Precise. **0b0** If the EDWAR is valid, it holds the virtual address of an access or sequence of contiguous accesses that triggered the Watchpoint debug event. **0b1** If the EDWAR is valid, it holds any virtual address within the smallest implemented translation granule that contains the virtual address of an access or set of contiguous accesses that triggered the Watchpoint debug event.	x
[14:11]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[10]	FnV	EDWAR not Valid. **0b0** EDWAR is valid. **0b1** EDWAR is not valid, and holds an **UNKNOWN** value.	x
[9:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
Debug	0x038	EDHSR	None

This interface is accessible as follows:

When DoubleLockStatus(), or !IsCorePowered() or OSLockStatus()
ERROR

Otherwise
RO

B.4.2 EDRCR, External Debug Reserve Control Register

This register is used to allow imprecise entry to Debug state and clear sticky bits in EDSCR.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
Debug

Register offset
0x090

Access type
WO

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-45: EXT_EDRCR bit assignments

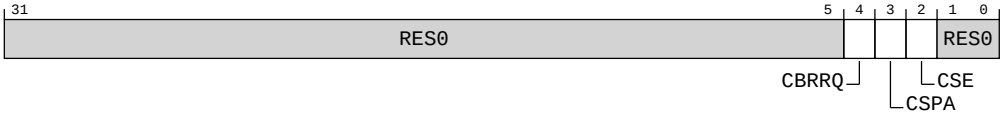


Table B-93: EDRCR bit descriptions

Bits	Name	Description	Reset
[31:5]	**RES0**	Reserved	**RES0**
[4]	CBRRQ	Allow imprecise entry to Debug state. The actions on writing to this bit are: **0b0** No action.	x
[3]	CSPA	Clear Sticky Pipeline Advance. This bit is used to clear the EDSCR.PipeAdv bit to 0. The actions on writing to this bit are: **0b0** No action. **0b1** Clear the EDSCR.PipeAdv bit to 0.	x
[2]	CSE	Clear Sticky Error. Used to clear the EDSCR cumulative error bits to 0. The actions on writing to this bit are: **0b0** No action. **0b1** Clear the EDSCR.{TXU, RXO, ERR} bits, and, if the PE is in Debug state, the EDSCR.ITO bit, to 0.	x
[1:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
Debug	0x090	EDRCR	None

This interface is accessible as follows:

When DoubleLockStatus(), or !IsCorePowered() or OSLockStatus()

ERROR

When SoftwareLockStatus()

WI

Otherwise

WO

B.4.3 EDPRCR, External Debug Power/Reset Control Register

Controls the PE functionality related to powerup, reset, and powerdown.

Configurations

CORENPDRQ is the only field that is mapped between the EDPRCR and DBGPRCR and DBGPRCR_EL1.

Attributes

Width

32

Component

Debug

Register offset

0x310

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx0x
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-46: EXT_EDPRCR bit assignments

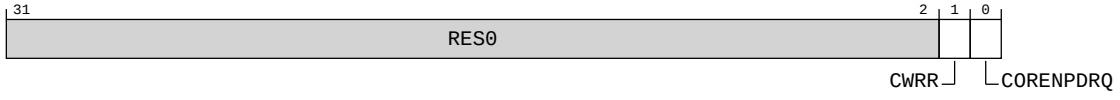


Table B-95: EDPRCR bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[1]	CWRR	Warm reset request. The extent of the reset is **IMPLEMENTATION DEFINED**, but must be one of: • The request is ignored. • Only this PE is Warm reset. • This PE and other components of the system, possibly including other PEs, are Warm reset. Arm deprecates use of this bit, and recommends that implementations ignore the request. **0b0** No action. **0b1** Request Warm reset. **When OSLockStatus() or SoftwareLockStatus()** Access to this field is: **RAZ/WI** **Otherwise** Access to this field is: WO/**RAZ**	0b0
[0]	CORENPDRQ	Core no powerdown request. Requests emulation of powerdown. This request is typically passed to an external power controller. This means that whether a request causes power up is dependent on the **IMPLEMENTATION DEFINED** nature of the system. The power controller must not allow the Core power domain to switch off while this bit is 1. **0b0** If the system responds to a powerdown request, it powers down Core power domain. **0b1** If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain. **When OSLockStatus()** Access to this field is: **UNKNOWN**/WI **When SoftwareLockStatus()** Access to this field is: RO **Otherwise** Access to this field is: RW	x [1]

Accessibility

On permitted accesses to the register, other access controls affect the behavior of some fields. See the field descriptions for more information.

Component	Offset	Instance	Range
Debug	0x310	EDPRCR	None

This interface is accessible as follows:

[1] On a Cold reset, if the powerup request is implemented and the powerup request has been asserted, this field is an **IMPLEMENTATION DEFINED** choice of 0 or 1. If the powerup request is not asserted, this field is set to 0.

When !IsCorePowered()

ERROR

When SoftwareLockStatus()

RO

Otherwise

RW

B.4.4 MIDR_EL1, Main ID Register

Provides identification information for the PE, including an implementer code for the device and a device ID number.

Configurations

External register MIDR_EL1 bits [31:0] are architecturally mapped to AArch64 System register A.5.1 MIDR_EL1, Main ID Register on page 401 bits [31:0].

Attributes

Width

32

Component

Debug

Register offset

0xD00

Access type

RO

Reset value

```
0100 0001 0001 1111 1101 1000 1100 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-47: EXT_MIDR_EL1 bit assignments

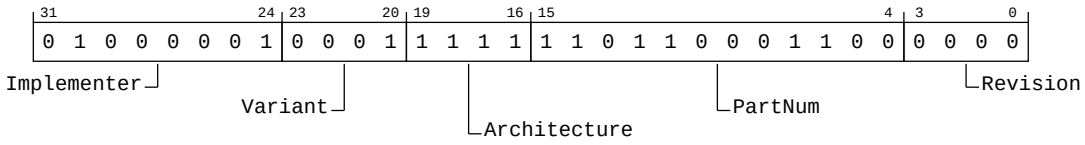


Table B-97: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	Implementer	Indicates the implementer code. This value is: **0x41** Arm Limited	0x41
[23:20]	Variant	Indicates the major revision of the product. **0b0001** r1p0	0b0001
[19:16]	Architecture	Architecture version. **0b1111** Architectural features are individually identified in the ID_* registers.	0b1111
[15:4]	PartNum	Primary Part Number for the device. On processors implemented by Arm, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently. **0xD8C** C1-Ultra	0xD8C
[3:0]	Revision	Indicates the minor revision of the product. **0b0000** r1p0	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xD00	MIDR_EL1	None

This interface is accessible as follows:

When DoubleLockStatus() or !IsCorePowered()

ImplementationDefined

Otherwise

RO

B.4.5 EDPFR, External Debug Processor Feature Register

Provides information about implemented PE features.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offset

0xD20

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx 0001 xxxx 0001 0001 xxxx xxxx 0001 0001 0001 0001 0001 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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_EDPFR bit assignments

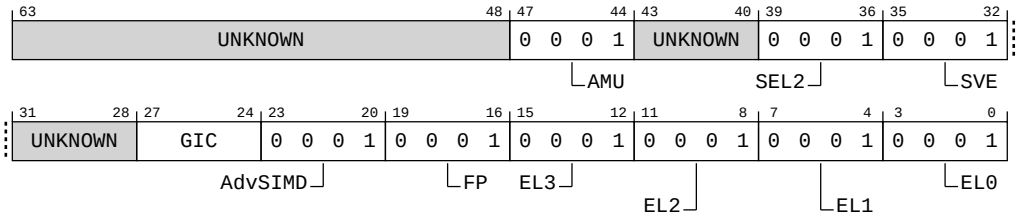


Table B-99: EDPFR bit descriptions

Bits	Name	Description	Reset
[63:48]	UNKNOWN	Reserved	UNKNOWN
[47:44]	AMU	Indicates support for Activity Monitors Extension. **0b0001** FEAT_AMUv1 is implemented.	0b0001
[43:40]	UNKNOWN	Reserved	UNKNOWN
[39:36]	SEL2	Secure EL2. **0b0001** Secure EL2 is implemented.	0b0001

Bits	Name	Description	Reset
[35:32]	SVE	Scalable Vector Extension. **0b0001** SVE is implemented.	0b0001
[31:28]	UNKNOWN	Reserved	UNKNOWN
[27:24]	GIC	System register GIC interface support. **0b0000** GIC CPU interface system registers not implemented. This value applies when GICCDISABLE == TRUE. **0b0011** System register interface to version 4.1 of the GIC CPU interface is supported. This value applies when GICCDISABLE == FALSE.	The reset values can be the following: 0b0000, 0b0011, respective to the value.
[23:20]	AdvSIMD	Advanced SIMD. **0b0001** As for 0b0000, and also includes support for half-precision floating-point arithmetic.	0b0001
[19:16]	FP	Floating-point. **0b0001** As for 0b0000, and also includes support for half-precision floating-point arithmetic.	0b0001
[15:12]	EL3	AArch64 EL3 Exception level handling. **0b0001** EL3 can be executed in AArch64 state only.	0b0001
[11:8]	EL2	AArch64 EL2 Exception level handling. **0b0001** EL2 can be executed in AArch64 state only.	0b0001
[7:4]	EL1	AArch64 EL1 Exception level handling. **0b0001** EL1 can be executed in AArch64 state only.	0b0001
[3:0]	EL0	AArch64 EL0 Exception level handling. **0b0001** EL0 can be executed in AArch64 state only.	0b0001

Accessibility

Component	Offset	Instance	Range
Debug	0xD20	EDPFR	None

This interface is accessible as follows:

When IsCorePowered() and !DoubleLockStatus()

RO

When !IsCorePowered()

ERROR

Otherwise

ImplementationDefined

B.4.6 EDDFR, External Debug Feature Register

Provides top level information about the debug system.

Note

Debuggers must use EDDEVARCH to determine the Debug architecture version.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offset

0xD28

Access type

RO

Reset value

```
xxxx 0000 xxxx xxxx xxxx 0001 xxxx xxxx 0001 0000 0011 0000 0101 1000 0001 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-49: EXT_EDDFR bit assignments

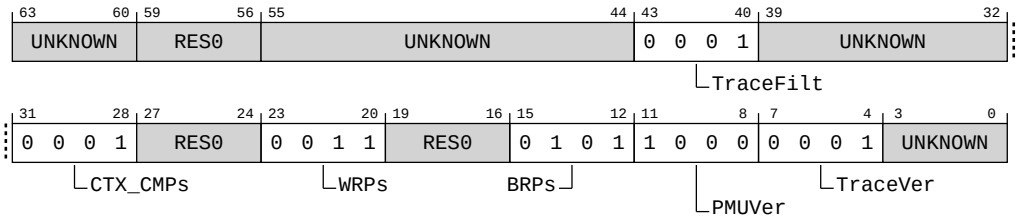


Table B-101: EDDFR bit descriptions

Bits	Name	Description	Reset
[63:60]	**UNKNOWN**	Reserved	**UNKNOWN**
[59:56]	**RES0**	Reserved	**RES0**
[55:44]	**UNKNOWN**	Reserved	**UNKNOWN**
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. **0b0001** Armv8.4 Self-hosted Trace Extension implemented.	0b0001
[39:32]	**UNKNOWN**	Reserved	**UNKNOWN**
[31:28]	CTX_CMPs	Number of context-aware breakpoints, minus 1. **0b0001** Two context-aware breakpoints are included	0b0001
[27:24]	**RES0**	Reserved	**RES0**
[23:20]	WRPs	Number of watchpoints, minus 1. **0b0011** Four watchpoints	0b0011
[19:16]	**RES0**	Reserved	**RES0**
[15:12]	BRPs	Number of breakpoints, minus 1. **0b0101** Six breakpoints	0b0101
[11:8]	PMUVer	Performance Monitors Extension version. This field does not follow the standard ID scheme, but uses the alternative ID scheme described in *Alternative ID scheme used for the Performance Monitors Extension version* in the *Arm® Architecture Reference Manual for A-profile architecture* **0b1000** PMUv3 for Armv8.8. As 0b0111, and: • Extends the Common event number space to include 0x0040 to 0x00BF and 0x4040 to 0x40BF. • Removes the **CONSTRAINED UNPREDICTABLE** behaviors if a reserved or unimplemented PMU event number is selected.	0b1000
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a trace unit is implemented. **0b0001** Trace unit System registers implemented.	0b0001

Bits	Name	Description	Reset
[3:0]	UNKNOWN	Reserved	UNKNOWN

Accessibility

Component	Offset	Instance	Range
Debug	0xD28	EDDFR	None

This interface is accessible as follows:

When IsCorePowered() and !DoubleLockStatus()

RO

When !IsCorePowered()

ERROR

Otherwise

ImplementationDefined

B.4.7 EDDFR1, External Debug Feature Register 1

Provides top level information about the debug system in AArch64.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offset

0xD48

Access type

RO

Reset value

```
0000 0000 xxxx 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
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-50: EXT_EDDFR1 bit assignments

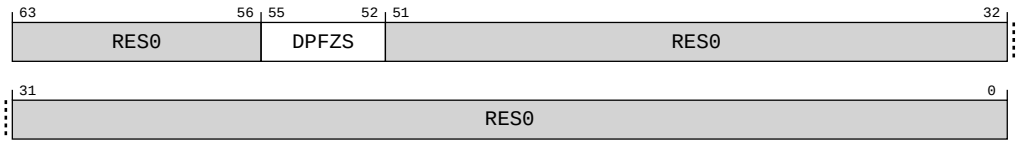


Table B-103: EDDFR1 bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[55:52]	DPFZS	This field either has the same value as ID_AA64DFR1_EL1.DPFZS or reads as zero. **0b0001..0b1111**	The reset values can be the following: `0b0001..0b1111`, respective to the value.
[51:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
Debug	0xD48	EDDFR1	None

This interface is accessible as follows:

When IsCorePowered(), !DoubleLockStatus() and (!IsZero(EDDFR1) or !OSLockStatus())

RO

When !IsCorePowered()

ERROR

Otherwise

ImplementationDefined

B.4.8 EDDEVARCH, External Debug Device Architecture Register

Identifies the programmers' model architecture of the external debug component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xFBC

Access type

RO

Reset value

```
0100 0111 0111 0000 1010 1010 0001 0101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-51: EXT_EDDEVARCH bit assignments

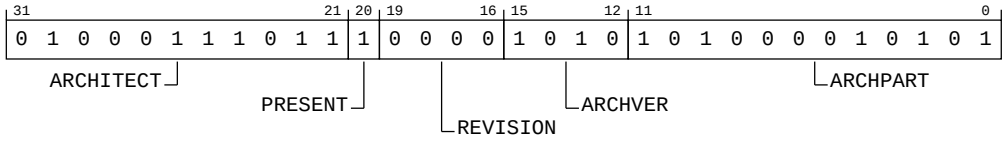


Table B-105: EDDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architect of the component. For External Debug, 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 EDDEVARCH register is present. **0b1**	`0b1`
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. For debug, the revision defined by Armv8 is `0x0`. All other values are reserved. **0b0000**	`0b0000`
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. Defined values are: **0b1010** Armv8.8 debug architecture, FEAT_Debugv8p8.	`0b1010`
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **0xA15** Armv8-A debug architecture.	`0xA15`

Accessibility

Component	Offset	Instance	Range
Debug	0xFBC	EDDEVARCH	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.4.9 EDDEVID2, External Debug Device ID register 2

Reserved for future descriptions of features of the debug implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xFC0

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-52: EXT_EDDEVID2 bit assignments

Table B-107: EDDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
Debug	0xFC0	EDDEVID2	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.4.10 EDDEVID1, External Debug Device ID Register 1

Provides extra information for external debuggers about features of the debug implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xFC4

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0001 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-53: EXT_EDDEVID1 bit assignments

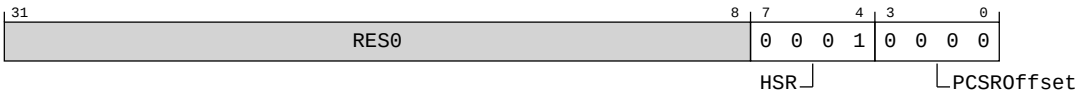


Table B-109: EDDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	HSR	Indicates support for the External Debug Halt Status Register, EDHSR. **0b0001** EDHSR implemented.	0b0001

Bits	Name	Description	Reset
[3:0]	PCSROffset	Indicates the offset applied to PC samples returned by reads of EDPCSR. **0b0000** EDPCSR not implemented.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFC4	EDDEVID1	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.4.11 EDDEVID, External Debug Device ID register 0

Provides extra information for external debuggers about features of the debug implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xFC8

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0001 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-54: EXT_EDDEVID bit assignments

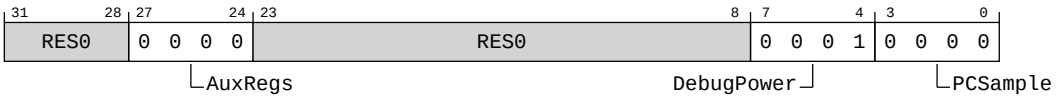


Table B-111: EDDEVID bit descriptions

Bits	Name	Description	Reset
[31:28]	**RES0**	Reserved	**RES0**
[27:24]	AuxRegs	Indicates support for Auxiliary registers. **0b0000** None supported.	0b0000
[23:8]	**RES0**	Reserved	**RES0**
[7:4]	DebugPower	Indicates support for the FEAT_DoPD feature. **0b0001** FEAT_DoPD implemented. All registers in the external debug interface register map are implemented in the Core power domain.	0b0001
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using external debug registers. **0b0000** PC Sample-based Profiling Extension is not implemented in the external debug registers space.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFC8	EDDEVID	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.4.12 EDDEVTYPE, External Debug Device Type register

Indicates to a debugger that this component is part of a PE's debug logic.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xFCC

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0001 0101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-55: EXT_EDDEVTYPE bit assignments

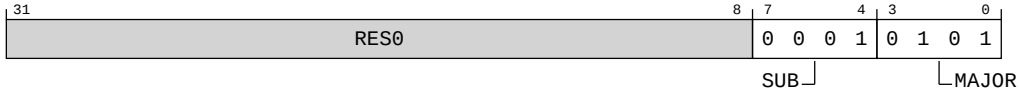


Table B-113: EDDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Subtype. Indicates this is a component within a PE. **0b0001**	0b0001
[3:0]	MAJOR	Major type. Indicates this is a debug logic component. **0b0101**	0b0101

Accessibility

Component	Offset	Instance	Range
Debug	0xFCC	EDDEVTYPE	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.4.13 EDPIDR4, External Debug Peripheral Identification Register 4

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

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-56: EXT_EDPIDR4 bit assignments

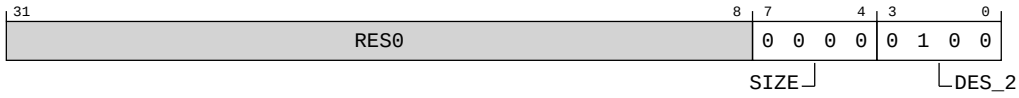


Table B-115: EDPIDR4 bit descriptions

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

Accessibility

Component	Offset	Instance	Range
Debug	0xFD0	EDPIDR4	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.4.14 EDPIDR0, External Debug Peripheral Identification Register 0

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1000 1100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-57: EXT_EDPIDR0 bit assignments

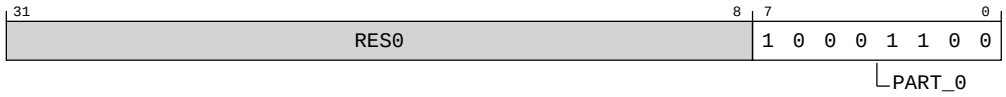


Table B-117: EDPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0x8C** C1-Ultra	0x8C

Accessibility

Component	Offset	Instance	Range
Debug	0xFE0	EDPIDR0	None

This interface is accessible as follows:

When !IsCorePowered()
ERROR

Otherwise
RO

B.4.15 EDPIDR1, External Debug Peripheral Identification Register 1

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width
32

Component
Debug

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-58: EXT_EDPIDR1 bit assignments

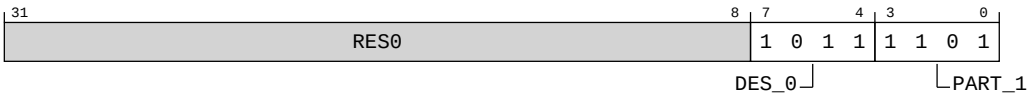


Table B-119: EDPIDR1 bit descriptions

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-Ultra	`0b1101`

Accessibility

Component	Offset	Instance	Range
Debug	0xFE4	EDPIDR1	None

This interface is accessible as follows:

When !IsCorePowered()
ERROR

Otherwise
RO

B.4.16 EDPIDR2, External Debug Peripheral Identification Register 2

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width
32

Component
Debug

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-59: EXT_EDPIDR2 bit assignments

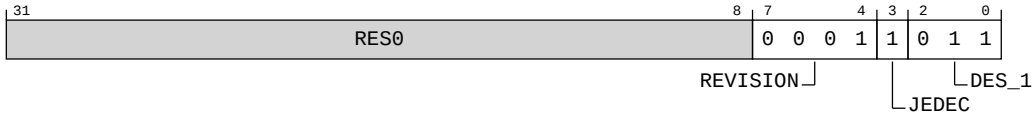


Table B-121: EDPIDR2 bit descriptions

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** r1p0	0b0001
[3]	JEDEC	Indicates a JEP106 identity code is used. **0b1**	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	Instance	Range
Debug	0xFE8	EDPIDR2	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.4.17 EDPIDR3, External Debug Peripheral Identification Register 3

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFEC

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-60: EXT_EDPIDR3 bit assignments

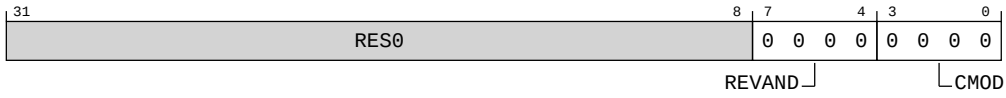


Table B-123: EDPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Part minor revision. Parts using EDPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0000** r1p0	0b0000
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFEC	EDPIDR3	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.4.18 EDCIDR0, External Debug Component Identification Register 0

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

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-61: EXT_EDCIDR0 bit assignments

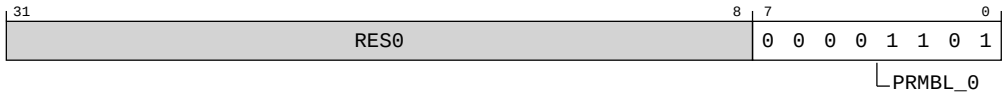


Table B-125: EDCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble. **0x0D**	0x0D

Accessibility

Component	Offset	Instance	Range
Debug	0xFF0	EDCIDR0	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.4.19 EDCIDR1, External Debug Component Identification Register 1

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

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-62: EXT_EDCIDR1 bit assignments

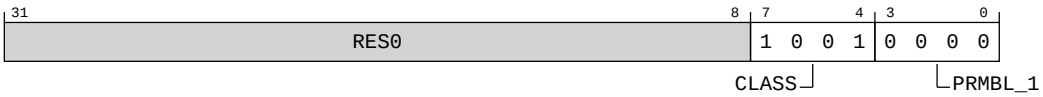


Table B-127: EDCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight component.	0b1001
[3:0]	PRMBL_1	Preamble. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFF4	EDCIDR1	None

This interface is accessible as follows:

When !IsCorePowered()
ERROR

Otherwise
RO

B.4.20 EDCIDR2, External Debug Component Identification Register 2

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width
32

Component
Debug

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-63: EXT_EDCIDR2 bit assignments

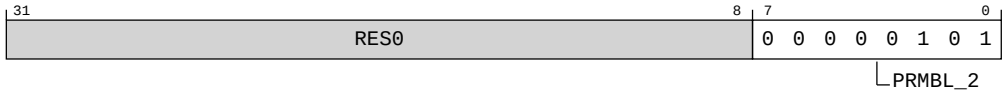


Table B-129: EDCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble. **0x05**	0x05

Accessibility

Component	Offset	Instance	Range
Debug	0xFF8	EDCIDR2	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.4.21 EDCIDR3, External Debug Component Identification Register 3

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

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-64: EXT_EDCIDR3 bit assignments

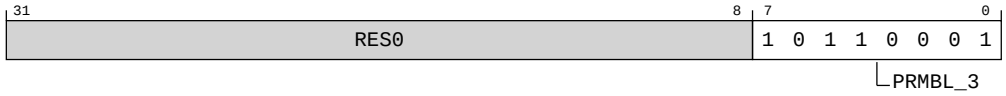


Table B-131: EDCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble. **0xB1**	0xB1

Accessibility

Component	Offset	Instance	Range
Debug	0xFFC	EDCIDR3	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.5 External ETE registers summary

The following summary table provides an overview of all memory-mapped ETE 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-133: ETE registers summary

Offset	Name	Reset	Width	Description
0x018	TRCAUXCTLR	See individual bit resets.	32-bit	Trace Auxiliary Control Register
0x180	TRCIDR8	0x00000000	32-bit	Trace ID Register 8
0x184	TRCIDR9	0x00000000	32-bit	Trace ID Register 9
0x188	TRCIDR10	0x00000000	32-bit	Trace ID Register 10
0x18C	TRCIDR11	0x00000000	32-bit	Trace ID Register 11
0x190	TRCIDR12	0x00000000	32-bit	Trace ID Register 12
0x194	TRCIDR13	0x00000000	32-bit	Trace ID Register 13
0x1C0	TRCIMSPEC0	See individual bit resets.	32-bit	Trace IMP DEF Register 0
0x1E0	TRCIDR0	0x28800EA1	32-bit	Trace ID Register 0
0x1E4	TRCIDR1	0x4100FFF0	32-bit	Trace ID Register 1
0x1E8	TRCIDR2	0xC0001088	32-bit	Trace ID Register 2
0x1EC	TRCIDR3	0x017F0004	32-bit	Trace ID Register 3
0x1F0	TRCIDR4	0x11170004	32-bit	Trace ID Register 4
0x1F4	TRCIDR5	0x284709FF	32-bit	Trace ID Register 5
0x1F8	TRCIDR6	0x00000000	32-bit	Trace ID Register 6
0x1FC	TRCIDR7	0x00000000	32-bit	Trace ID Register 7
0xF00	TRCITCTRL	See individual bit resets.	32-bit	Trace Integration Mode Control Register
0xFA0	TRCCLAIMSET	0xFFFFFFFF	32-bit	Trace Claim Tag Set Register
0xFA4	TRCCLAIMCLR	0x00000000	32-bit	Trace Claim Tag Clear Register
0xFBC	TRCDEVARCH	0x47715A13	32-bit	Trace Device Architecture Register
0xFC0	TRCDEVID2	0x00000000	32-bit	Trace Device Configuration Register 2
0xFC4	TRCDEVID1	0x00000000	32-bit	Trace Device Configuration Register 1
0xFC8	TRCDEVID	0x00000000	32-bit	Trace Device Configuration Register
0xFCC	TRCDEVTYPE	0x00000013	32-bit	Trace Device Type Register
0xFD0	TRCPIDR4	See individual bit resets.	32-bit	Trace Peripheral Identification Register 4
0xFD4	TRCPIDR5	0x00000000	32-bit	Trace Peripheral Identification Register 5
0xFD8	TRCPIDR6	0x00000000	32-bit	Trace Peripheral Identification Register 6
0xFDC	TRCPIDR7	0x00000000	32-bit	Trace Peripheral Identification Register 7
0xFE0	TRCPIDR0	0x0000008C	32-bit	Trace Peripheral Identification Register 0
0xFE4	TRCPIDR1	0x000000BD	32-bit	Trace Peripheral Identification Register 1

Offset	Name	Reset	Width	Description
0xFE8	TRCPIDR2	0x0000001B	32-bit	Trace Peripheral Identification Register 2
0xFEC	TRCPIDR3	0x00000000	32-bit	Trace Peripheral Identification Register 3
0xFF0	TRCCIDR0	0x0000000D	32-bit	Trace Component Identification Register 0
0xFF4	TRCCIDR1	0x00000090	32-bit	Trace Component Identification Register 1
0xFF8	TRCCIDR2	0x00000005	32-bit	Trace Component Identification Register 2
0xFFC	TRCCIDR3	0x000000B1	32-bit	Trace Component Identification Register 3

B.5.1 TRCAUXCTLR, Trace Auxiliary Control Register

The function of this register is **IMPLEMENTATION DEFINED**.

Configurations

External register TRCAUXCTLR bits [31:0] are architecturally mapped to AArch64 System register TRCAUXCTLR bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x018

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-65: EXT_TRCAUXCTLR bit assignments

Table B-134: TRCAUXCTLR bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

If this register is nonzero then it might cause the behavior of a trace unit to contradict this architecture specification. See the documentation of the specific implementation for information about the **IMPLEMENTATION DEFINED** support for this register.

Component	Offset	Instance	Range
ETE	0x018	TRCAUXCTLR	None

This interface is accessible as follows:

When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered()

ERROR

Otherwise

RW

B.5.2 TRCIDR8, Trace ID Register 8

Returns the maximum speculation depth of the instruction trace element stream.

Configurations

External register TRCIDR8 bits [31:0] are architecturally mapped to AArch64 System register A.16.1 TRCIDR8, Trace ID Register 8 on page 594 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x180

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-66: EXT_TRCIDR8 bit assignments

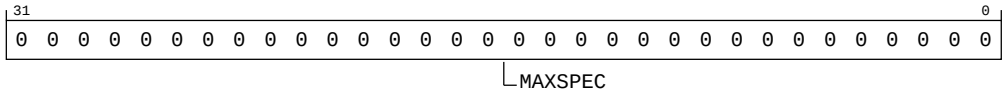


Table B-136: TRCIDR8 bit descriptions

Bits	Name	Description	Reset
[31:0]	MAXSPEC	Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of P0 elements in the trace element stream that can be speculative at any time. **0x00000000** No speculation in the trace element stream	0x00000000

Accessibility

Component	Offset	Instance	Range
ETE	0x180	TRCIDR8	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.3 TRCIDR9, Trace ID Register 9

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR9 bits [31:0] are architecturally mapped to AArch64 System register TRCIDR9 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x184

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-67: EXT_TRCIDR9 bit assignments

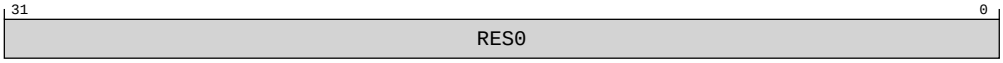


Table B-138: TRCIDR9 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x184	TRCIDR9	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.4 TRCIDR10, Trace ID Register 10

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR10 bits [31:0] are architecturally mapped to AArch64 System register A.16.3 TRCIDR10, Trace ID Register 10 on page 598 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x188

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-68: EXT_TRCIDR10 bit assignments

Table B-140: TRCIDR10 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x188	TRCIDR10	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.5 TRCIDR11, Trace ID Register 11

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR11 bits [31:0] are architecturally mapped to AArch64 System register A.16.4 TRCIDR11, Trace ID Register 11 on page 600 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x18C

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-69: EXT_TRCIDR11 bit assignments

Table B-142: TRCIDR11 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x18C	TRCIDR11	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.6 TRCIDR12, Trace ID Register 12

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR12 bits [31:0] are architecturally mapped to AArch64 System register A.16.5 TRCIDR12, Trace ID Register 12 on page 601 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x190

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-70: EXT_TRCIDR12 bit assignments

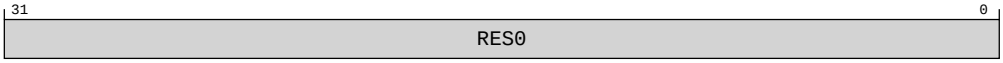


Table B-144: TRCIDR12 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x190	TRCIDR12	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.7 TRCIDR13, Trace ID Register 13

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR13 bits [31:0] are architecturally mapped to AArch64 System register A.16.6 TRCIDR13, Trace ID Register 13 on page 603 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x194

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-71: EXT_TRCIDR13 bit assignments

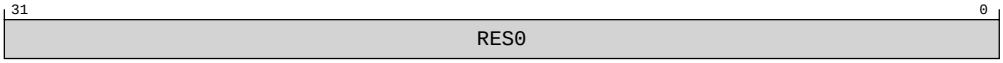


Table B-146: TRCIDR13 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x194	TRCIDR13	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.8 TRCIMSPEC0, Trace IMP DEF Register 0

TRCIMSPEC0 shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

Configurations

External register TRCIMSPEC0 bits [31:0] are architecturally mapped to AArch64 System register A.16.2 TRCIMSPEC0, Trace IMP DEF Register 0 on page 596 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1C0

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-72: EXT_TRCIMSPEC0 bit assignments

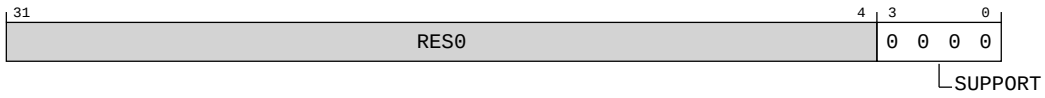


Table B-148: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[31:4]	**RES0**	Reserved	**RES0**
[3:0]	SUPPORT	Indicates whether the implementation supports **IMPLEMENTATION DEFINED** features. **0b0000** No **IMPLEMENTATION DEFINED** features are supported.	0b0000

Accessibility

Component	Offset	Instance	Range
ETE	0x1C0	TRCIMSPEC0	None

This interface is accessible as follows:

When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered()

ERROR

Otherwise

RW

B.5.9 TRCIDR0, Trace ID Register 0

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR0 bits [31:0] are architecturally mapped to AArch64 System register A.16.7 TRCIDR0, Trace ID Register 0 on page 604 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1E0

Access type

RO

Reset value

```
0010 1000 1000 0000 0000 1110 1010 0001
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-73: EXT_TRCIDR0 bit assignments

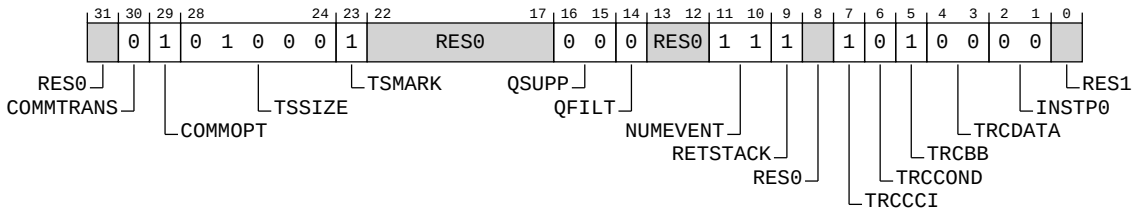


Table B-150: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31]	**RES0**	Reserved	**RES0**
[30]	COMMTRANS	Transaction Start element behavior. **0b0** Transaction Start elements are P0 elements.	0b0
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **0b1** Commit mode 1. Access to this field is: **RAO/WI**	0b1
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. **0b01000** Global timestamping implemented with a 64-bit timestamp value.	0b01000
[23]	TSMARK	Indicates whether Timestamp Marker elements are generated. **0b1** Timestamp Marker elements are generated.	0b1
[22:17]	**RES0**	Reserved	**RES0**
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **0b00** Q element support is not implemented.	0b00
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	0b0
[13:12]	**RES0**	Reserved	**RES0**
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **0b11** The trace unit supports 4 ETEEvents.	0b11
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **0b1** Return stack implemented.	0b1
[8]	**RES0**	Reserved	**RES0**
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **0b1** Cycle counting implemented.	0b1
[6]	TRCCOND	Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b0** Conditional instruction tracing not implemented.	0b0
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. **0b1** Branch broadcasting implemented.	0b1

Bits	Name	Description	Reset
[4:3]	TRCDATA	Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Tracing of data addresses and data values is not implemented.	0b00
[2:1]	INSTP0	Indicates if load and store instructions are P0 instructions. Load and store instructions as P0 instructions is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Load and store instructions are not P0 instructions.	0b00
[0]	**RES1**	Reserved	**RES1**

Accessibility

Component	Offset	Instance	Range
ETE	0x1E0	TRCIDR0	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()
ERROR

Otherwise
RO

B.5.10 TRCIDR1, Trace ID Register 1

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR1 bits [31:0] are architecturally mapped to AArch64 System register A.16.8 TRCIDR1, Trace ID Register 1 on page 607 bits [31:0].

Attributes

Width
32

Component
ETE

Register offset
0x1E4

Access type
RO

Reset value

```
0100 0001 0000 0000 1111 1111 1111 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-74: EXT_TRCIDR1 bit assignments

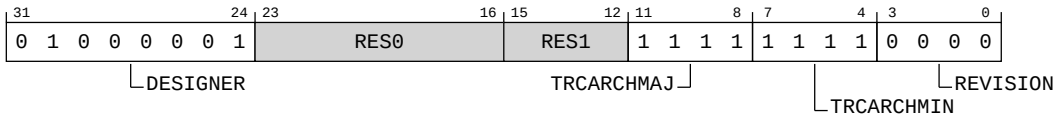


Table B-152: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:24]	DESIGNER	Indicates which company designed the trace unit. The permitted values of this field are the same as MIDR_EL1.Implementer. **0x41** Arm Limited	0x41
[23:16]	**RES0**	Reserved	**RES0**
[15:12]	**RES1**	Reserved	**RES1**
[11:8]	TRCARCHMAJ	Major architecture version. **0b1111** If both TRCIDR1.TRCARCHMAJ and TRCIDR1.TRCARCHMIN == 0xF then refer to TRCDEVARCH.	0b1111
[7:4]	TRCARCHMIN	Minor architecture version. **0b1111** If both TRCIDR1.TRCARCHMAJ and TRCIDR1.TRCARCHMIN == 0xF then refer to TRCDEVARCH.	0b1111
[3:0]	REVISION	Implementation revision. Returns an **IMPLEMENTATION DEFINED** value that identifies the revision of the trace unit. Arm deprecates any use of this field and recommends that implementations set this field to zero. **0b0000** Revision 0	0b0000

Accessibility

Component	Offset	Instance	Range
ETE	0x1E4	TRCIDR1	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.11 TRCIDR2, Trace ID Register 2

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR2 bits [31:0] are architecturally mapped to AArch64 System register A.16.9 TRCIDR2, Trace ID Register 2 on page 609 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1E8

Access type

RO

Reset value

```
1100 0000 0000 0000 0001 0000 1000 1000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-75: EXT_TRCIDR2 bit assignments

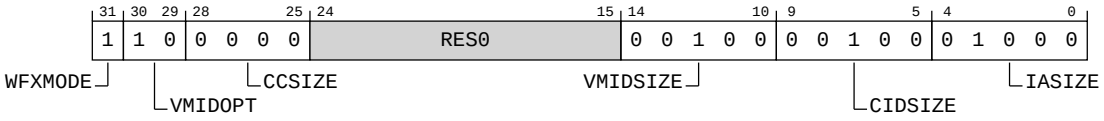


Table B-154: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31]	WFXMODE	Indicates whether `WFI`, `WFIT`, `WFE`, and `WFET` instructions are classified as P0 instructions: **0b1** `WFI`, `WFIT`, `WFE`, and `WFET` instructions are classified as P0 instructions.	0b1
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. **0b10** Virtual context identifier selection not supported. TRCCONFIGR.VMIDOPT is **RES1**.	0b10
[28:25]	CCSIZE	Indicates the size of the cycle counter. **0b0000** The cycle counter is 12 bits in length.	0b0000
[24:15]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. **0b00100** 32-bit Virtual context identifier size.	0b00100
[9:5]	CIDSIZE	Indicates the Context identifier size. **0b00100** 32-bit Context identifier size.	0b00100
[4:0]	IASIZE	Virtual instruction address size. **0b01000** Maximum of 64-bit instruction address size.	0b01000

Accessibility

Component	Offset	Instance	Range
ETE	0x1E8	TRCIDR2	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()
ERROR

Otherwise
RO

B.5.12 TRCIDR3, Trace ID Register 3

Returns the base architecture of the trace unit.

Configurations

External register TRCIDR3 bits [31:0] are architecturally mapped to AArch64 System register A.16.10 TRCIDR3, Trace ID Register 3 on page 611 bits [31:0].

Attributes

Width
32

Component
ETE

Register offset
0x1EC

Access type
RO

Reset value

```
0000 0001 0111 1111 0000 0000 0000 0100
|    |    |    |    |    |    |    |  |
```

```
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-76: EXT_TRCIDR3 bit assignments

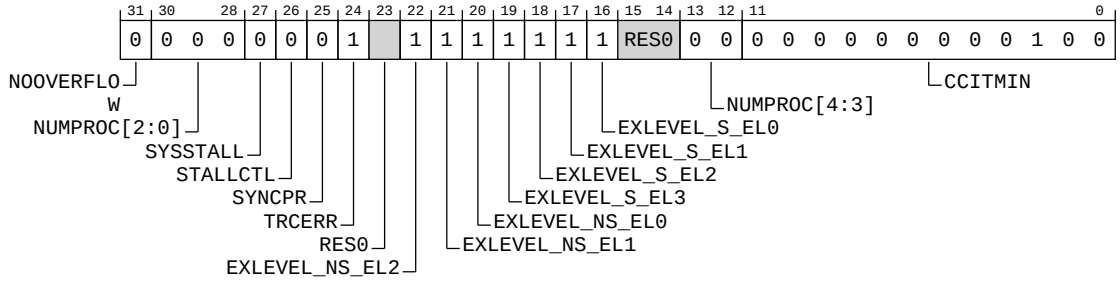


Table B-156: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	0b0
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **0b0** Stalling of the PE is not permitted.	0b0
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **0b0** Stalling of the PE is not implemented.	0b0
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0b0** TRCSYNCPR is read/write so software can change the synchronization period.	0b0
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **0b1** Forced tracing of System Error exceptions is implemented.	0b1
[23]	**RES0**	Reserved	**RES0**
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 is implemented. **0b1** Non-secure EL2 is implemented.	0b1
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 is implemented. **0b1** Non-secure EL1 is implemented.	0b1
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 is implemented. **0b1** Non-secure EL0 is implemented.	0b1

Bits	Name	Description	Reset
[19]	EXLEVEL_S_EL3	Indicates if EL3 is implemented. **0b1** EL3 is implemented.	0b1
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 is implemented. **0b1** Secure EL2 is implemented.	0b1
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 is implemented. **0b1** Secure EL1 is implemented.	0b1
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 is implemented. **0b1** Secure EL0 is implemented.	0b1
[15:14]	**RES0**	Reserved	**RES0**
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. **0b00000** The trace unit can trace one PE.	0b00000
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in TRCCCCTLR.THRESHOLD. **0x004** The minimum value that can be programmed in TRCCCCTLR.THRESHOLD.	0x004

Accessibility

Component	Offset	Instance	Range
ETE	0x1EC	TRCIDR3	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.13 TRCIDR4, Trace ID Register 4

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR4 bits [31:0] are architecturally mapped to AArch64 System register A.16.11 TRCIDR4, Trace ID Register 4 on page 614 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1F0

Access type

RO

Reset value

```
0001 0001 0001 0111 0000 0000 0000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-77: EXT_TRCIDR4 bit assignments

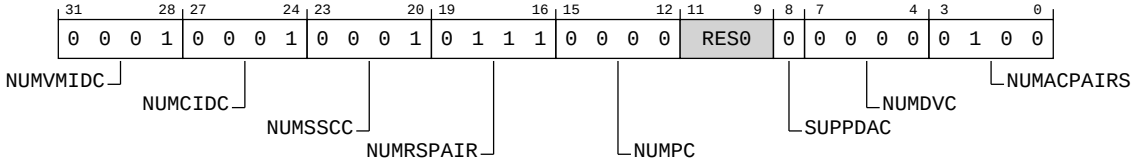


Table B-158: TRCIDR4 bit descriptions

Bits	Name	Description	Reset
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. **0b0001** The number of Virtual Context Identifier Comparators in this implementation.	0b0001
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **0b0001** The number of Context Identifier Comparators in this implementation.	0b0001
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. **0b0001** The number of Single-shot Comparator Controls in this implementation.	0b0001
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0b0111** The number of resource selector pairs in this implementation, minus one.	0b0111
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0b0000** The number of PE Comparator Inputs in this implementation.	0b0000
[11:9]	**RES0**	Reserved	**RES0**
[8]	SUPPDAC	Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0** Data address comparisons not implemented.	0b0

Bits	Name	Description	Reset
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0000** The number of data value comparators in this implementation.	0b0000
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **0b0100** The number of Address Comparator pairs in this implementation.	0b0100

Accessibility

Component	Offset	Instance	Range
ETE	0x1F0	TRCIDR4	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()
ERROR

Otherwise
RO

B.5.14 TRCIDR5, Trace ID Register 5

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR5 bits [31:0] are architecturally mapped to AArch64 System register A.16.12 TRCIDR5, Trace ID Register 5 on page 616 bits [31:0].

Attributes

Width
32

Component
ETE

Register offset
0x1F4

Access type
RO

Reset value

```
0010 1000 0100 0111 0000 1001 1111 1111
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-78: EXT_TRCIDR5 bit assignments

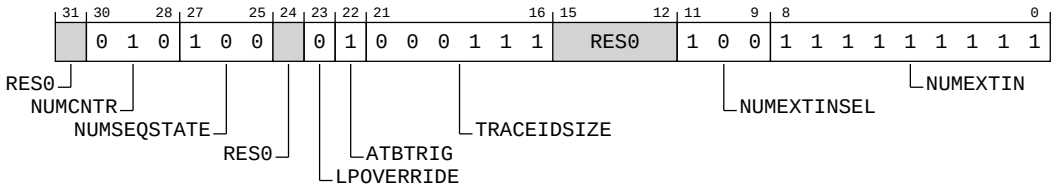


Table B-160: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[31]	**RES0**	Reserved	**RES0**
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. **0b010** The number of Counters implemented.	0b010
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **0b100** Four Sequencer states are implemented.	0b100
[24]	**RES0**	Reserved	**RES0**
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **0b0** The trace unit does not support Low-power Override Mode.	0b0
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **0b1** The implementation supports ATB triggers.	0b1
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **0b000111** The implementation supports a 7-bit trace ID.	0b000111
[15:12]	**RES0**	Reserved	**RES0**
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. **0b100** The number of External Input Selector resources implemented.	0b100
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **0b111111111** Unified PMU event selection.	0b111111111

Accessibility

Component	Offset	Instance	Range
ETE	0x1F4	TRCIDR5	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.15 TRCIDR6, Trace ID Register 6

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR6 bits [31:0] are architecturally mapped to AArch64 System register TRCIDR6 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1F8

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-79: EXT_TRCIDR6 bit assignments

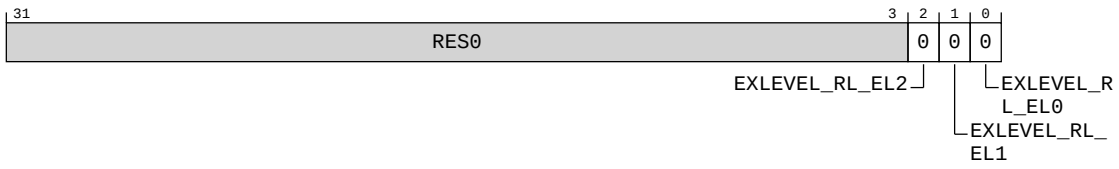


Table B-162: TRCIDR6 bit descriptions

Bits	Name	Description	Reset
[31:3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2]	EXLEVEL_RL_EL2	Indicates if Realm EL2 is implemented. **0b0** Realm EL2 is not implemented.	0b0
[1]	EXLEVEL_RL_EL1	Indicates if Realm EL1 is implemented. **0b0** Realm EL1 is not implemented.	0b0
[0]	EXLEVEL_RL_EL0	Indicates if Realm EL0 is implemented. **0b0** Realm EL0 is not implemented.	0b0

Accessibility

Component	Offset	Instance	Range
ETE	0x1F8	TRCIDR6	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.16 TRCIDR7, Trace ID Register 7

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR7 bits [31:0] are architecturally mapped to AArch64 System register TRCIDR7 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1FC

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-80: EXT_TRCIDR7 bit assignments

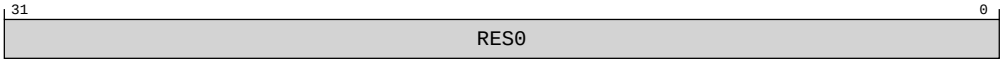


Table B-164: TRCIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
ETE	0x1FC	TRCIDR7	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.17 TRCITCTRL, Trace Integration Mode Control Register

A component can use TRCITCTRL to dynamically switch between functional mode and integration mode. In integration mode, topology detection is enabled. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xF00

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-81: EXT_TRCITCTRL bit assignments

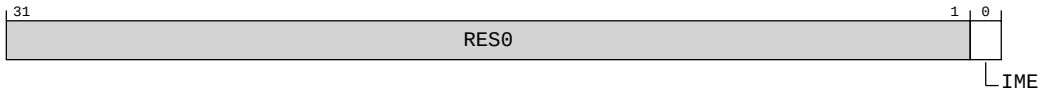


Table B-166: TRCITCTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	**RES0**	Reserved	**RES0**
[0]	IME	Integration Mode Enable. **0b0** Component functional mode. **0b1** Component integration mode. Support for topology detection and integration testing is enabled.	x

Accessibility

External debugger accesses to this register are **IMPLEMENTATION DEFINED** when the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0xF00	TRCITCTRL	None

This interface is accessible as follows:

When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered()

ERROR

Otherwise

RW

B.5.18 TRCCLAIMSET, Trace Claim Tag Set Register

In conjunction with TRCCLAIMCLR, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information, see the CoreSight Architecture Specification.

Configurations

The number of claim tag bits implemented is **IMPLEMENTATION DEFINED**. Arm recommends that implementations support a minimum of four claim tag bits, that is, SET[3:0] reads as `0b1111`.

External register TRCCLAIMSET bits [31:0] are architecturally mapped to AArch64 System register TRCCLAIMSET bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0xFA0

Access type

RW

Reset value

```
1111 1111 1111 1111 1111 1111 1111 1111
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-82: EXT_TRCCLAIMSET bit assignments

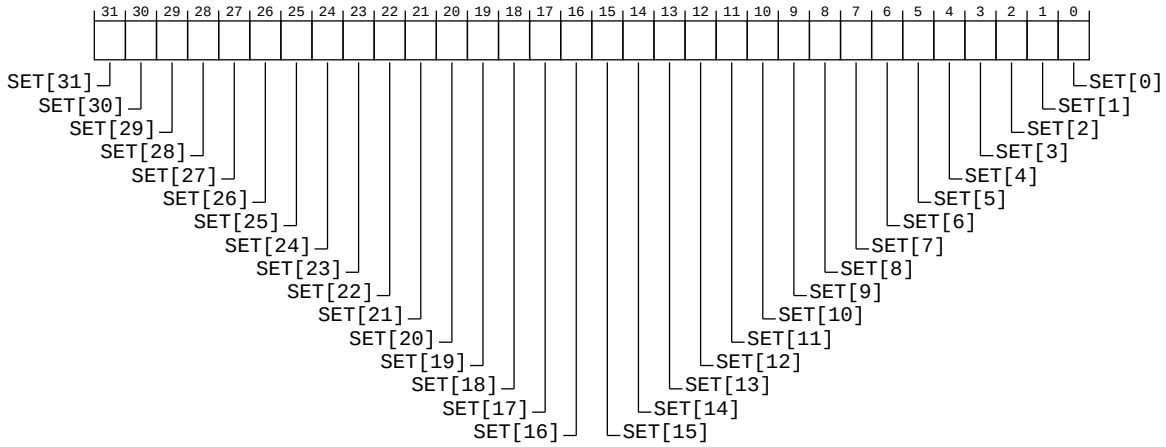


Table B-168: TRCCLAIMSET bit descriptions

Bits	Name	Description	Reset
[31:0]	SET[<m>], bit[m], where m = 31 to 0	Claim Tag Set. Indicates whether Claim Tag bit <m> is implemented, and is used to set Claim Tag bit <m> to 1. **0b0** On a read: Claim Tag bit <m> is not implemented. On a write: Ignored. **0b1** On a read: Claim Tag bit <m> is implemented. On a write: Set Claim Tag bit <m> to 1.	0xFFFFFFFF

Accessibility

Component	Offset	Instance	Range
ETE	0xFA0	TRCCLAIMSET	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RW

B.5.19 TRCCLAIMCLR, Trace Claim Tag Clear Register

In conjunction with TRCCLAIMSET, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information, see the CoreSight Architecture Specification.

Configurations

External register TRCCLAIMCLR bits [31:0] are architecturally mapped to AArch64 System register TRCCLAIMCLR bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0xFA4

Access type

RW

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-83: EXT_TRCCLAIMCLR bit assignments

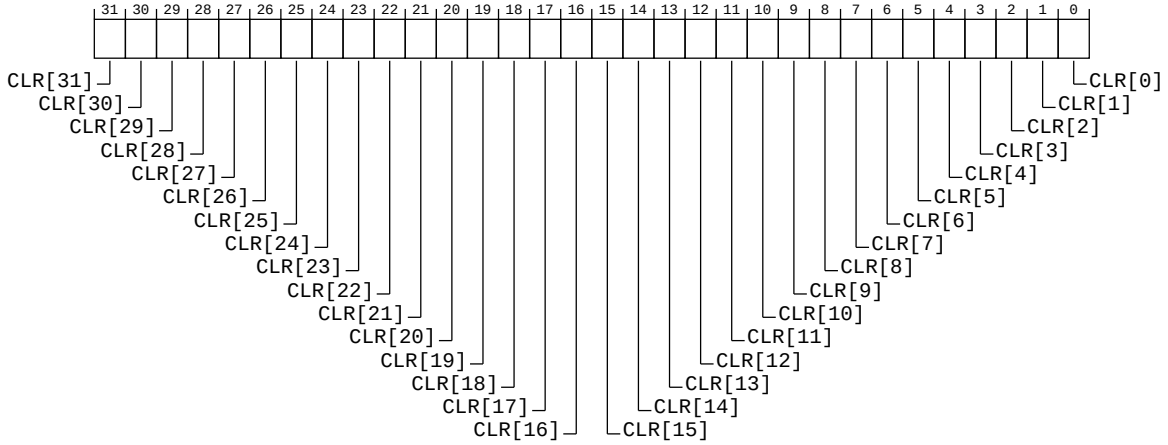


Table B-170: TRCCLAIMCLR bit descriptions

Bits	Name	Description	Reset
[31:0]	CLR[<m>], bit[m], where m = 31 to 0	Claim Tag Clear. Indicates the current status of Claim Tag bit <m>, and is used to clear Claim Tag bit <m> to 0. **0b0** On a read: Claim Tag bit <m> is not set. On a write: Ignored. **0b1** On a read: Claim Tag bit <m> is set. On a write: Clear Claim tag bit <m> to 0.	0x00000000

Accessibility

Component	Offset	Instance	Range
ETE	0xFA4	TRCCLAIMCLR	None

This interface is accessible as follows:

When OSLockStatus() or !IsTraceCorePowered()

ERROR

Otherwise

RW

B.5.20 TRCDEVARCH, Trace Device Architecture Register

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

External register TRCDEVARCH bits [31:0] are architecturally mapped to AArch64 System register TRCDEVARCH bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0xFBC

Access type

RO

Reset value

```
0100 0111 0111 0001 0101 1010 0001 0011
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-84: EXT_TRCDEVARCH bit assignments

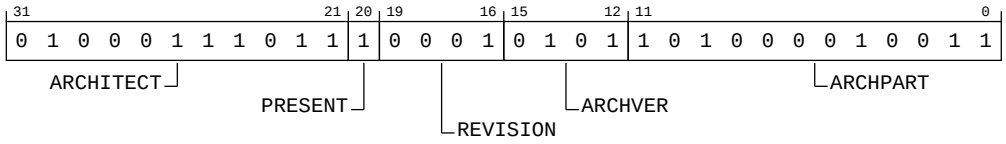


Table B-172: TRCDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architect of the component. For Trace, 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 TRCDEVARCH register is present. **`0b1`**	`0b1`

Bits	Name	Description	Reset
[19:16]	REVISION	Revision. Defines the architecture revision of the component. **0b0001** ETEv1.1, FEAT_ETEv1p1.	0b0001
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **0b0101** ETEv1.	0b0101
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **0xA13** Arm PE trace architecture.	0xA13

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFBC	TRCDEVARCH	None

This interface is accessible as follows:

When !IsTraceCorePowered()
ERROR

Otherwise
RO

B.5.21 TRCDEVID2, Trace Device Configuration Register 2

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFC0

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-85: EXT_TRCDEVID2 bit assignments

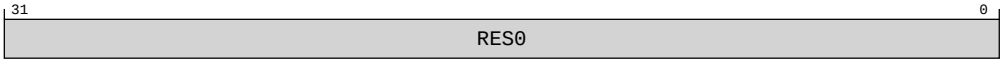


Table B-174: TRCDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFC0	TRCDEVID2	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.22 TRCDEVID1, Trace Device Configuration Register 1

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFC4

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-86: EXT_TRCDEVID1 bit assignments

Table B-176: TRCDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFC4	TRCDEVID1	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.23 TRCDEVID, Trace Device Configuration Register

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

External register TRCDEVID bits [31:0] are architecturally mapped to AArch64 System register TRCDEVID bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0xFC8

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-87: EXT_TRCDEVID bit assignments

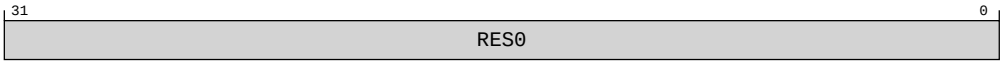


Table B-178: TRCDEVID bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFC8	TRCDEVID	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.24 TRCDEVTYPE, Trace Device Type Register

Provides discovery information for the component. If the part number field is not recognized, a debugger can report the information that is provided by TRCDEVTYPE about the component instead.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFCC

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0001 0011
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-88: EXT_TRCDEVTYPE bit assignments

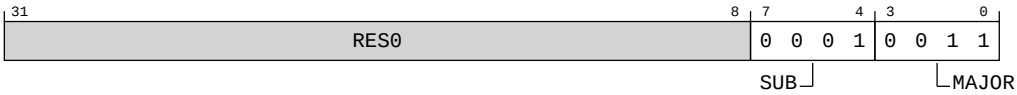


Table B-180: TRCDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Component sub-type. **0b0001** When MAJOR == 0x3 (Trace source): Associated with a PE.	0b0001
[3:0]	MAJOR	Component major type. **0b0011** Trace source.	0b0011

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFCC	TRCDEVTYPE	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.25 TRCPIDR4, Trace Peripheral Identification Register 4

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 xxxx 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-89: EXT_TRCPIDR4 bit assignments

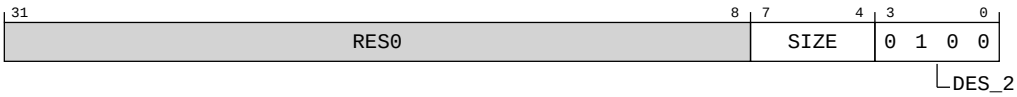


Table B-182: TRCPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	Size of the component. **0b0000** The component uses a single 4KB block **0b0001..0b1111** The component occupies $2^{TRCPIDR4.SIZE}$ 4KB blocks.	The reset values can be the following: 0b0000, 0b0001..0b1111, respective to the value.
[3:0]	DES_2	Designer, JEP106 continuation code. JEP106 identification and continuation codes, which are stored as follows: • TRCPIDR1.DES_0: JEP106 identification code bits[3:0]. • TRCPIDR2.DES_1: JEP106 identification code bits[6:4]. • TRCPIDR4.DES_2: JEP106 continuation code. These codes indicate the designer of the component and not the implementer, except where the two are the same. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. A JEDEC code takes the following form: • A sequence of zero or more numbers, all having the value 0x7F. • A following 8-bit number, that is not 0x7F, and where bit[7] is an odd parity bit. The parity bit in the JEP106 identification code is not included. **0b0100** Arm Limited	0b0100

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFD0	TRCPIDR4	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.26 TRCPIDR5, Trace Peripheral Identification Register 5

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD4

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-90: EXT_TRCPIDR5 bit assignments

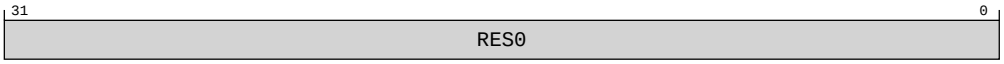


Table B-184: TRCPIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFD4	TRCPIDR5	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise
RO

B.5.27 TRCPIDR6, Trace Peripheral Identification Register 6

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFD8

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-91: EXT_TRCPIDR6 bit assignments

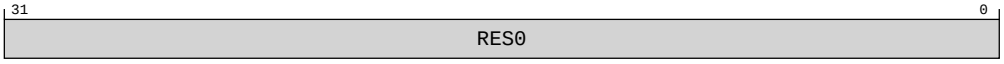


Table B-186: TRCPIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFD8	TRCPIDR6	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.28 TRCPIDR7, Trace Peripheral Identification Register 7

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFDC

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-92: EXT_TRCPIDR7 bit assignments

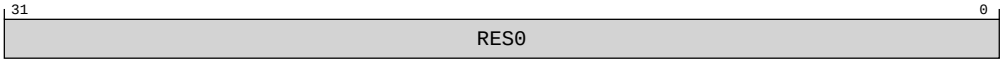


Table B-188: TRCPIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFDC	TRCPIDR7	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.29 TRCPIDR0, Trace Peripheral Identification Register 0

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1000 1100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-93: EXT_TRCPIDR0 bit assignments

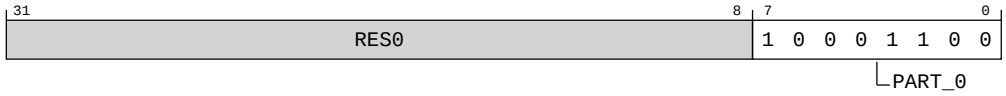


Table B-190: TRCPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, bits [7:0]. The part number is selected by the designer of the component, and is stored in TRCPIDR1.PART_1 and TRCPIDR0.PART_0. **0x8C** C1-Ultra	0x8C

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFE0	TRCPIDR0	None

This interface is accessible as follows:

When !IsTraceCorePowered()
ERROR

Otherwise
RO

B.5.30 TRCPIDR1, Trace Peripheral Identification Register 1

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFE4

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-94: EXT_TRCPIDR1 bit assignments

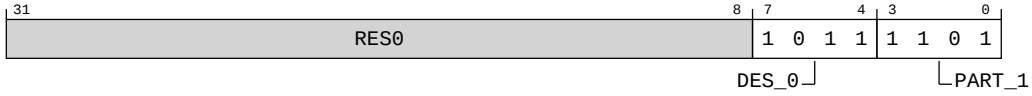


Table B-192: TRCPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, JEP106 identification code, bits [3:0]. JEP106 identification and continuation codes, which are stored as follows: • TRCPIDR1.DES_0: JEP106 identification code bits[3:0]. • TRCPIDR2.DES_1: JEP106 identification code bits[6:4]. • TRCPIDR4.DES_2: JEP106 continuation code. These codes indicate the designer of the component and not the implementer, except where the two are the same. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. A JEDEC code takes the following form: • A sequence of zero or more numbers, all having the value `0x7F`. • A following 8-bit number, that is not `0x7F`, and where bit[7] is an odd parity bit. The parity bit in the JEP106 identification code is not included. **`0b1011`** Arm Limited	`0b1011`
[3:0]	PART_1	Part number, bits [11:8]. The part number is selected by the designer of the component, and is stored in TRCPIDR1.PART_1 and TRCPIDR0.PART_0. **`0b1101`** C1-Ultra	`0b1101`

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFE4	TRCPIDR1	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise
RO

B.5.31 TRCPIDR2, Trace Peripheral Identification Register 2

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFE8

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-95: EXT_TRCPIDR2 bit assignments

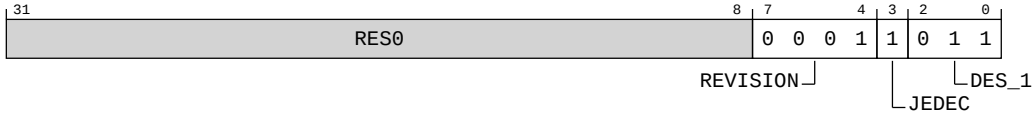


Table B-194: TRCPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	REVISION	Component major revision. TRCPIDR2.REVISION and TRCPIDR3.REVAND together form the revision number of the component, with TRCPIDR2.REVISION being the most significant part and TRCPIDR3.REVAND the least significant part. When a component is changed, TRCPIDR2.REVISION or TRCPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. TRCPIDR3.REVAND should be set to `0b0000` when TRCPIDR2.REVISION is increased. **`0b0001`** r1p0	`0b0001`
[3]	JEDEC	JEDEC-assigned JEP106 implementer code is used. **`0b1`**	`0b1`
[2:0]	DES_1	Designer, JEP106 identification code, bits [6:4]. JEP106 identification and continuation codes, which are stored as follows: • TRCPIDR1.DES_0: JEP106 identification code bits[3:0]. • TRCPIDR2.DES_1: JEP106 identification code bits[6:4]. • TRCPIDR4.DES_2: JEP106 continuation code. These codes indicate the designer of the component and not the implementer, except where the two are the same. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. A JEDEC code takes the following form: • A sequence of zero or more numbers, all having the value `0x7F`. • A following 8-bit number, that is not `0x7F`, and where bit[7] is an odd parity bit. The parity bit in the JEP106 identification code is not included. **`0b011`** Arm Limited	`0b011`

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFE8	TRCPIDR2	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.32 TRCPIDR3, Trace Peripheral Identification Register 3

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFEC

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-96: EXT_TRCPIDR3 bit assignments

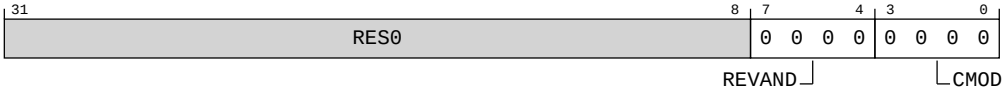


Table B-196: TRCPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Component minor revision. TRCPIDR2.REVISION and TRCPIDR3.REVAND together form the revision number of the component, with TRCPIDR2.REVISION being the most significant part and TRCPIDR3.REVAND the least significant part. When a component is changed, TRCPIDR2.REVISION or TRCPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. TRCPIDR3.REVAND should be set to `0b0000` when TRCPIDR2.REVISION is increased. **`0b0000`** r1p0	`0b0000`

Bits	Name	Description	Reset
[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

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFEC	TRCPIDR3	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.33 TRCCIDR0, Trace Component Identification Register 0

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF0

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-97: EXT_TRCCIDR0 bit assignments

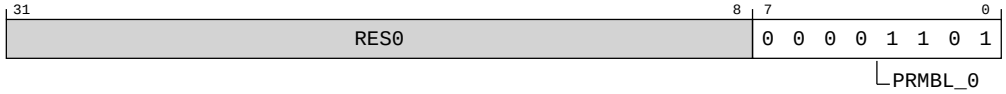


Table B-198: TRCCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Component identification preamble, segment 0. **0x0D**	0x0D

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFF0	TRCCIDR0	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.34 TRCCIDR1, Trace Component Identification Register 1

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF4

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-98: EXT_TRCCIDR1 bit assignments

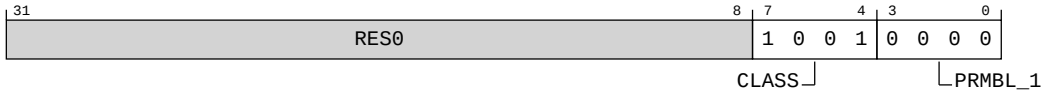


Table B-200: TRCCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight peripheral.	0b1001
[3:0]	PRMBL_1	Component identification preamble, segment 1. **0b0000**	0b0000

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFF4	TRCCIDR1	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.35 TRCCIDR2, Trace Component Identification Register 2

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF8

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-99: EXT_TRCCIDR2 bit assignments

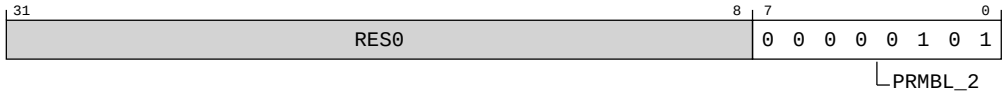


Table B-202: TRCCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Component identification preamble, segment 2. **0x05**	0x05

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFF8	TRCCIDR2	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.36 TRCCIDR3, Trace Component Identification Register 3

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFFC

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-100: EXT_TRCCIDR3 bit assignments

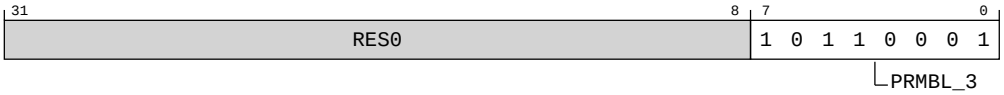


Table B-204: TRCCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Component identification preamble, segment 3. **0xB1**	0xB1

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFFC	TRCCIDR3	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.6 External PMU registers summary

The following summary table provides an overview of all memory-mapped PMU registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the *Arm® Architecture Reference Manual for A-profile architecture*.

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-206: PMU registers summary

Offset	Name	Reset	Width	Description
0x0	PMEVCNTR0_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x8	PMEVCNTR1_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x10	PMEVCNTR2_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x18	PMEVCNTR3_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x20	PMEVCNTR4_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x28	PMEVCNTR5_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x30	PMEVCNTR6_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x38	PMEVCNTR7_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x40	PMEVCNTR8_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x48	PMEVCNTR9_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x50	PMEVCNTR10_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x58	PMEVCNTR11_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x60	PMEVCNTR12_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x68	PMEVCNTR13_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x70	PMEVCNTR14_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x78	PMEVCNTR15_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x80	PMEVCNTR16_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x88	PMEVCNTR17_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x90	PMEVCNTR18_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x98	PMEVCNTR19_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers

Offset	Name	Reset	Width	Description
0xA0	PMEVCNTR20_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xA8	PMEVCNTR21_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xB0	PMEVCNTR22_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xB8	PMEVCNTR23_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xC0	PMEVCNTR24_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xC8	PMEVCNTR25_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xD0	PMEVCNTR26_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xD8	PMEVCNTR27_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xE0	PMEVCNTR28_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xE8	PMEVCNTR29_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0xF0	PMEVCNTR30_EL0 [63:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
0x0F8	PMCCNTR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter
0x0FC	PMCCNTR_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter
0x200	PMPCSR [31:0]	See individual bit resets.	32-bit	Program Counter Sample Register
0x204	PMPCSR [63:32]	See individual bit resets.	32-bit	Program Counter Sample Register
0x220	PMPCSR [31:0]	See individual bit resets.	32-bit	Program Counter Sample Register
0x224	PMPCSR [63:32]	See individual bit resets.	32-bit	Program Counter Sample Register
0x208	PMCID1SR	See individual bit resets.	32-bit	CONTEXTIDR_EL1 Sample Register
0x228	PMCID1SR	See individual bit resets.	32-bit	CONTEXTIDR_EL1 Sample Register
0x20C	PMVIDSR	See individual bit resets.	32-bit	VMID Sample Register
0x22C	PMCID2SR	See individual bit resets.	32-bit	CONTEXTIDR_EL2 Sample Register
0x400	PMEVTYPER0_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA00	PMEVTYPER0_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x404	PMEVTYPER1_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA04	PMEVTYPER1_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x408	PMEVTYPER2_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA08	PMEVTYPER2_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x40C	PMEVTYPER3_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA0C	PMEVTYPER3_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x410	PMEVTYPER4_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA10	PMEVTYPER4_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x414	PMEVTYPER5_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA14	PMEVTYPER5_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x418	PMEVTYPER6_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA18	PMEVTYPER6_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x41C	PMEVTYPER7_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA1C	PMEVTYPER7_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x420	PMEVTYPER8_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA20	PMEVTYPER8_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x424	PMEVTYPER9_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers

Offset	Name	Reset	Width	Description
0xA24	PMEVTYPER9_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x428	PMEVTYPER10_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA28	PMEVTYPER10_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x42C	PMEVTYPER11_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA2C	PMEVTYPER11_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x430	PMEVTYPER12_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA30	PMEVTYPER12_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x434	PMEVTYPER13_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA34	PMEVTYPER13_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x438	PMEVTYPER14_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA38	PMEVTYPER14_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x43C	PMEVTYPER15_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA3C	PMEVTYPER15_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x440	PMEVTYPER16_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA40	PMEVTYPER16_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x444	PMEVTYPER17_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA44	PMEVTYPER17_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x448	PMEVTYPER18_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA48	PMEVTYPER18_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x44C	PMEVTYPER19_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA4C	PMEVTYPER19_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x450	PMEVTYPER20_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA50	PMEVTYPER20_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x454	PMEVTYPER21_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA54	PMEVTYPER21_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x458	PMEVTYPER22_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA58	PMEVTYPER22_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x45C	PMEVTYPER23_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA5C	PMEVTYPER23_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x460	PMEVTYPER24_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA60	PMEVTYPER24_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x464	PMEVTYPER25_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA64	PMEVTYPER25_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x468	PMEVTYPER26_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA68	PMEVTYPER26_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x46C	PMEVTYPER27_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA6C	PMEVTYPER27_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x470	PMEVTYPER28_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA70	PMEVTYPER28_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x474	PMEVTYPER29_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers

Offset	Name	Reset	Width	Description
0xA74	PMEVTYPER29_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x478	PMEVTYPER30_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0xA78	PMEVTYPER30_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x47C	PMCCFILTR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter Filter Register
0xA7C	PMCCFILTR_EL0 [63:32]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter Filter Register
0x600	PMPCSSR	See individual bit resets.	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x60C	PMCID2SSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register
0x610	PMSSSR	See individual bit resets.	32-bit	PMU Snapshot Status Register
0x618	PMCCNTSR	See individual bit resets.	64-bit	PMU Cycle Counter Snapshot Register
0x620	PMEVCNTSR0	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x628	PMEVCNTSR1	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x630	PMEVCNTSR2	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x638	PMEVCNTSR3	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x640	PMEVCNTSR4	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x648	PMEVCNTSR5	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x650	PMEVCNTSR6	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x658	PMEVCNTSR7	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x660	PMEVCNTSR8	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x668	PMEVCNTSR9	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x670	PMEVCNTSR10	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x678	PMEVCNTSR11	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x680	PMEVCNTSR12	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x688	PMEVCNTSR13	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x690	PMEVCNTSR14	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x698	PMEVCNTSR15	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A0	PMEVCNTSR16	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A8	PMEVCNTSR17	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B0	PMEVCNTSR18	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B8	PMEVCNTSR19	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6C0	PMEVCNTSR20	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6C8	PMEVCNTSR21	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6D0	PMEVCNTSR22	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6D8	PMEVCNTSR23	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6E0	PMEVCNTSR24	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6E8	PMEVCNTSR25	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F0	PMEVCNTSR26	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F8	PMEVCNTSR27	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x700	PMEVCNTSR28	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x708	PMEVCNTSR29	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register

Offset	Name	Reset	Width	Description
0x710	PMEVCNTSR30	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0xC00	PMCNTENSET_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Count Enable Set Register
0xC20	PMCNTENCLR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Count Enable Clear Register
0xC40	PMINTENSET_EL1 [31:0]	See individual bit resets.	32-bit	Performance Monitors Interrupt Enable Set Register
0xC60	PMINTENCLR_EL1 [31:0]	See individual bit resets.	32-bit	Performance Monitors Interrupt Enable Clear Register
0xC80	PMOVSCLR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Overflow Flag Status Clear register
0xCA0	PMSWINC_EL0	See individual bit resets.	32-bit	Performance Monitors Software Increment Register
0xCC0	PMOVSSET_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Overflow Flag Status Set Register
0xE00	PMCFGR [31:0]	See individual bit resets.	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_EL0	See individual bit resets.	32-bit	Performance Monitors Control Register
0xE08	PMIIDR	See individual bit resets.	32-bit	Performance Monitors Implementation Identification Register
0xE20	PMCEID0	0x7FFF6F3F	32-bit	Performance Monitors Common Event Identification register 0
0xE24	PMCEID1	0xFEF2AE7F	32-bit	Performance Monitors Common Event Identification register 1
0xE28	PMCEID2	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 2
0xE2C	PMCEID3	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 3
0xE30	PMSSCR	See individual bit resets.	32-bit	PMU Snapshot Capture Register
0xE40	PMMIR [31:0]	0x0000000A	32-bit	Performance Monitors Machine Identification Register
0xFA8	PMDEVAFF0	See individual bit resets.	32-bit	Performance Monitors Device Affinity register 0
0xFAC	PMDEVAFF1	See individual bit resets.	32-bit	Performance Monitors Device Affinity register 1
0xFB0	PMLAR	See individual bit resets.	32-bit	Performance Monitors Lock Access Register
0xFB4	PMLSR	See individual bit resets.	32-bit	Performance Monitors Lock Status Register
0xFB8	PMAUTHSTATUS	See individual bit resets.	32-bit	Performance Monitors Authentication Status register
0xFBC	PMDEVARCH	0x47702A16	32-bit	Performance Monitors Device Architecture register
0xFC8	PMDEVID	0x00000001	32-bit	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	0x00000016	32-bit	Performance Monitors Device Type register
0xFD0	PMPIDR4	0x00000004	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	0x0000008C	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	0x000000BD	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	0x0000001B	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	0x00000000	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	0x0000000D	32-bit	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	0x00000090	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	0x00000005	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	0x000000B1	32-bit	Performance Monitors Component Identification Register 3

B.6.1 PMPCSSR, Snapshot Program Counter Sample Register

Captured copy of the Program Counter.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x600

Access type

RO

Reset value

```
xxx0 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.

Bit descriptions

Figure B-101: PMU_PMPCSSR bit assignments

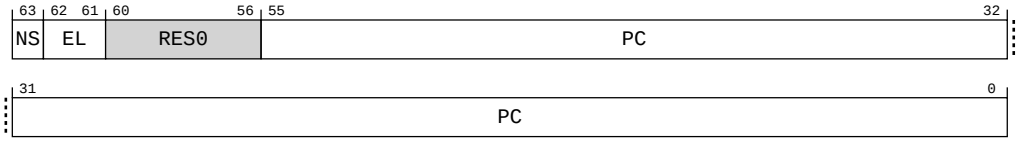


Table B-207: PMPCSSR bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure sample. **0b0** The captured instruction was executed in Secure state. **0b1** The captured instruction was executed in Non-secure state.	x

Bits	Name	Description	Reset
[62:61]	EL	Exception level sample. The Exception level the captured instruction was executed at.	xx
[60:56]	**RES0**	Reserved	**RES0**
[55:0]	PC	Sampled PC. The instruction address for the sampled instruction. The sampled instruction must be an instruction recently executed by the PE. The architecture does not require that all instructions are eligible for sampling. The sampled instruction must be architecturally executed. However, in exceptional circumstances, such as a change in security state or other boundary condition, it is permissible to sample an instruction that was speculatively executed and not architecturally executed.	56{x}

Accessibility

PMPCSSR is set to an **UNKNOWN** value by a read of the EDPCSRlo or PMPCSR[31:0] register.

Component	Offset	Instance	Range
PMU	0x600	PMPCSSR	None

This interface is accessible as follows:

RO

B.6.2 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register

Captured copy of the CONTEXTIDR_EL1 register.

The value captured must relate to the instruction captured in PMPCSSR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x608

Access type

RO

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-102: PMU_PMCIDSSR bit assignments

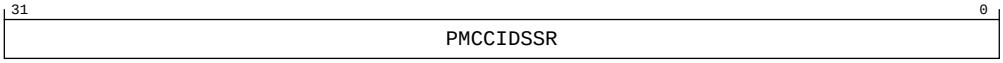


Table B-209: PMCIDSSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMCCIDSSR	PMCIDSR sample. Sampled CONTEXTIDR_EL1 snapshot.	32{x}

Accessibility

PMCIDSSR is set to an **UNKNOWN** value by a read of the EDPCSRlo or PMPCSR[31:0] register.

Component	Offset	Instance	Range
PMU	0x608	PMCIDSSR	None

This interface is accessible as follows:

RO

B.6.3 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register

Captured copy of the CONTEXTIDR_EL2 register.

The value captured must relate to the instruction captured in PMPCSSR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x60C

Access type

RO

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-103: PMU_PMCID2SSR bit assignments

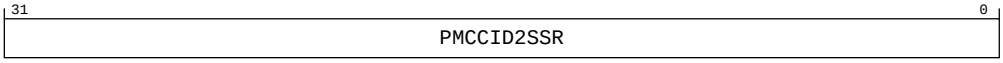


Table B-211: PMCID2SSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMCCID2SSR	PMCID2SR sample. Sampled CONTEXTIDR_EL2 snapshot.	32{x}

Accessibility

If ARMv8.2 is not implemented, this location is used for PMVIDSSR.

PMCID2SSR is set to an **UNKNOWN** value by a read of the EDPCSRlo or PMPCSR[31:0] register.

Component	Offset	Instance	Range
PMU	0x60C	PMCID2SSR	None

This interface is accessible as follows:

RO

B.6.4 PMSSSR, PMU Snapshot Status Register

Holds status information about the captured counters.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x610

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 000x
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-104: PMU_PMSSSR bit assignments

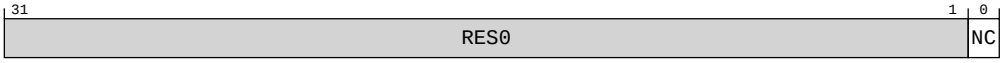


Table B-213: PMSSSR bit descriptions

Bits	Name	Description	Reset
[31:1]	**RES0**	Reserved	**RES0**
[0]	NC	No capture. Indicates whether the PMU counters have been captured. **0b0** PMU counters captured. **0b1** PMU counters not captured.	x

Accessibility

Component	Offset	Instance	Range
PMU	0x610	PMSSSR	None

This interface is accessible as follows:

RO

B.6.5 PMCCNTSR, PMU Cycle Counter Snapshot Register

Captured copy of PMCCNTR_EL0. Once captured, the value in PMCCNTSR is unaffected by writes to PMCCNTR_EL0 and PMCR_EL0.C.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x618

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.

Bit descriptions

Figure B-105: PMU_PMCCNTSR bit assignments

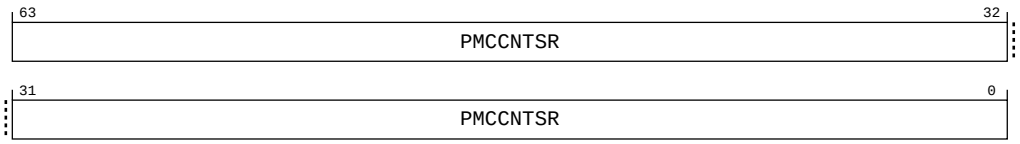


Table B-215: PMCCNTSR bit descriptions

Bits	Name	Description	Reset
[63:0]	PMCCNTSR	PMCCNTR_EL0 sample. Sampled cycle count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x618	PMCCNTSR	None

This interface is accessible as follows:

RO

B.6.6 PMCFGR, Performance Monitors Configuration Register

Contains PMU-specific configuration data.

Configurations

PMCFGR is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xE00

Access type

RO

Reset value

```
0000 0000 0x10 0000 0111 1111 xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-106: PMU_PMCFGR bit assignments

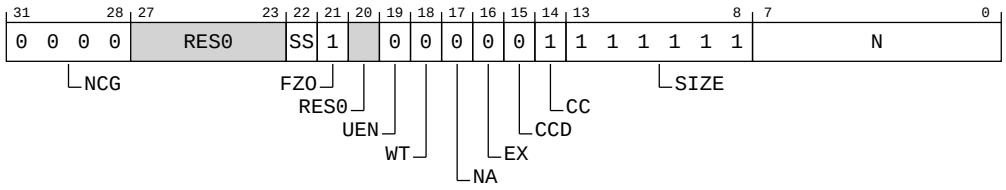


Table B-217: PMCFGR bit descriptions

Bits	Name	Description	Reset
[31:28]	NCG	Defines the number of counter groups implemented, minus one. **0b0000** One counter group implemented.	0b0000
[27:23]	**RES0**	Reserved	**RES0**
[22]	SS	Snapshot supported. **0b0** Snapshot mechanism not supported. The locations 0x600-0x7FC and 0xE30-0xE3C are **IMPLEMENTATION DEFINED**. **0b1** Snapshot mechanism supported. Locations 0x600-0x7FC and 0xE30-0xE3C contain **IMPLEMENTATION DEFINED** snapshot registers.	The reset values can be the following: 0b0, 0b1, respective to the value.
[21]	FZO	Freeze-on-overflow supported. **0b1** Freeze-on-overflow mechanism is supported. PMCR_EL0.FZO is RW.	0b1
[20]	**RES0**	Reserved	**RES0**
[19]	UEN	User-mode Enable Register supported. PMUSERENR_EL0 is not visible in the external debug interface, so this bit is **RAZ**. **0b0**	0b0
[18]	WT	This feature is not supported, so this bit is **RAZ**. **0b0**	0b0
[17]	NA	This feature is not supported, so this bit is **RAZ**. **0b0**	0b0
[16]	EX	Export supported. **0b0** PMCR_EL0.X is **RES0**.	0b0
[15]	CCD	Cycle counter has prescale. This field is **RAZ** **0b0** PMCR_EL0.D is **RES0**.	0b0

Bits	Name	Description	Reset
[14]	CC	Dedicated cycle counter (counter 31) supported. **0b1**	0b1
[13:8]	SIZE	Size of counters, minus one. This field defines the size of the largest counter implemented by the Performance Monitors Unit. From Armv8.0, the largest counter is 64-bits, so the value of this field is 0b111111. This field is used by software to determine the spacing of the counters in the memory-map. From Armv8.0, the counters are a doubleword-aligned addresses. **0b111111**	0b111111
[7:0]	N	Number of counters implemented, minus one. **0x06** Six event counters and the cycle counter implemented **0x1F** Thirty-one event counters and the cycle counter implemented	The reset values can be the following: 0x06, 0x1F, respective to the value.

Accessibility

Note

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE00	PMCFGR	31:0

This interface is accessible as follows:

When DoubleLockStatus(), or !IsCorePowered() or !AllowExternalPMUAccess()
ERROR

When OSLockStatus()
ERROR

Otherwise
RO

B.6.7 PMCR_EL0, Performance Monitors Control Register

Configures and controls the Performance Monitors counters.

Configurations

PMCR_EL0 is in the Core power domain.

External register PMCR_EL0 bits [31:0] are architecturally mapped to AArch64 System register A.8.2 PMCR_EL0, Performance Monitors Control Register on page 476 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xE04

Access type

RW

Reset value

```
0000 0000 0000 0000 0000 0xxx xxxx x000
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-107: PMU_PMCR_EL0 bit assignments

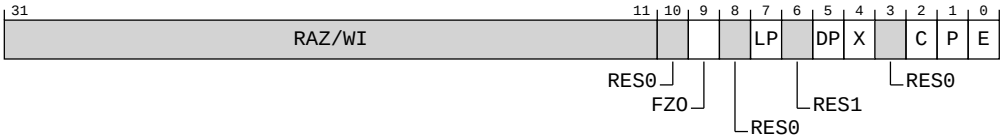


Table B-219: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[31:11]	**RAZ/WI**	Reserved	**RAZ/WI**
[10]	**RES0**	Reserved	**RES0**
[9]	FZO	Freeze-on-overflow. Stop event counters on overflow. **0b0** Do not freeze on overflow. **0b1** Affected counters do not count when PMOVSCLR_EL0[m] is 1 for any event counter PMEVCNTR<m>_EL0 in the first range.	x
[8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7]	LP	Long event counter enable. Determines when unsigned overflow is recorded by PMOVSCLR_EL0.P[n]. **0b0** Event counter overflow on increment that causes unsigned overflow of PMEVCNTR<n>_EL0[31:0]. **0b1** Event counter overflow on increment that causes unsigned overflow of PMEVCNTR<n>_EL0[63:0].	x
[6]	**RES1**	Reserved	**RES1**
[5]	DP	Disable cycle counter when event counting is prohibited. **0b0** Cycle counting by PMCCNTR_EL0 is not affected by this mechanism. **0b1** Cycle counting by PMCCNTR_EL0 is disabled in prohibited regions and when event counting is frozen: • If FEAT_PMUv3p1 is implemented, EL2 is implemented, and MDCR_EL2.HPMD or HDCR.HPMD is 1, then cycle counting by PMCCNTR_EL0 is disabled at EL2. • If FEAT_SPE_DPFZS is implemented and event counting is frozen by PMCR_EL0.FZS, then cycle counting by PMCCNTR_EL0 is disabled. • If FEAT_PMUv3p7 is implemented and event counting is frozen by PMCR_EL0.FZO, then cycle counting by PMCCNTR_EL0 is disabled. • If FEAT_PMUv3p7 is implemented, EL3 is implemented and using AArch64, and MDCR_EL3.MPMX is 1, then cycle counting by PMCCNTR_EL0 is disabled at EL3. • If EL3 is implemented, MDCR_EL3.SPME or SDCR.SPME is 0, and one of FEAT_PMUv3p7 is not implemented, EL3 is using AArch32, or MDCR_EL3.MPMX is 0, then cycle counting by PMCCNTR_EL0 is disabled at EL3 and in Secure state.	x
[4]	X	**When ImpDefBool("the implementation includes a PMU event export bus** Enable export of events in an **IMPLEMENTATION DEFINED** PMU event export bus. **0b0** Do not export events. **0b1** Export events where not prohibited. **Otherwise** RAZ/WI	xxxx
[3]	**RES0**	Reserved	**RES0**
[2]	C	Cycle counter reset. The effects of writing to this field are: **0b0** No action. **0b1** Reset PMCCNTR_EL0 to zero. Access to this field is: WO/**RAZ**	0b0

Bits	Name	Description	Reset
[1]	P	Event counter reset. **0b0** No action. **0b1** Reset all event counters PMEVCNTR<n>_EL0 to zero. Access to this field is: WO/**RAZ**	0b0
[0]	E	Enable. **0b0** Affected counters are disabled and do not count. **0b1** Affected counters are enabled by PMCNTENSET_EL0.	0b0

Accessibility

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE04	PMCR_EL0	None

This interface is accessible as follows:

When DoubleLockStatus(), or !IsCorePowered() or !AllowExternalPMUAccess()
ERROR

When OSLockStatus()
ERROR

When SoftwareLockStatus()
RO

Otherwise
RW

B.6.8 PMIIDR, Performance Monitors Implementation Identification Register

Provides discovery information about the Performance Monitor component.

Configurations

This register is present only when ImpDefBool("IMPLEMENTED_PMIIDR. Otherwise, direct accesses to PMIIDR are RES0.

Attributes

Width

32

Component

PMU

Register offset

0xE08

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0xxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-108: PMU_PMIIDR bit assignments

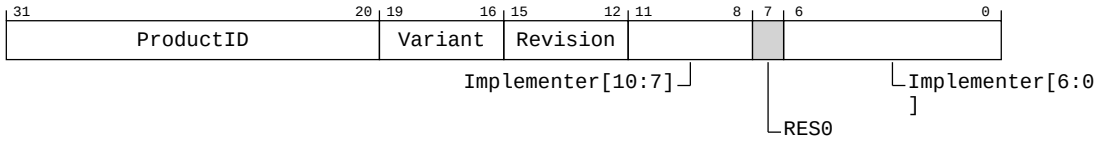


Table B-221: PMIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	Part number, bits [11:0]. The part number is selected by the designer of the component. Matches the PMPIDR1.PART_1, PMPIDR0.PART_0 fields if present.	12{x}

Bits	Name	Description	Reset
[19:16]	Variant	Component major revision. Defines either a variant of the component defined by PMIIDR.ProductID, or the major revision of the component. When defining a major revision, PMIIDR.Variant and PMIIDR.Revision together form the revision number of the component, with this field being the most significant part. When a component is changed, PMIIDR.Variant or PMIIDR.Revision is increased to ensure that software can differentiate between different revisions of the component. If this field is increased, PMIIDR.Revision should be set to `0b0000`. Matches the PMPIDR2.REVISION field, if present.	`xxxx`
[15:12]	Revision	Component minor revision. PMIIDR.Variant and PMIIDR.Revision together form the revision number of the component, with this field being the least significant part. When a component is changed, PMIIDR.Variant or PMIIDR.Revision is increased to ensure that software can differentiate between different revisions of the component. If PMIIDR.Variant field is increased, this field should be set to `0b0000`, otherwise the value in this field should be increased. Matches the PMPIDR3.REVAND field, if present.	`xxxx`
[11:8, 6:0]	Implementer	JEDEC-assigned JEP106 identifcation code of the designer of the component. Bits [11:8] are the JEP106 bank identifier minus 1 and bits [6:0] are the JEP106 identification code for the designer of the component. **Note:** For Arm Limited, the JEP106 bank is 5 and the JEP106 identification code is `0x3B`, meaning PMIIDR[11:0] has the value `0x43B`. Bits [11:8] match the PMPIDR4.DES_2 field, if present. Bits[6:0] match the {PMPIDR2.DES1, PMPIDR1.DES_0} fields if present.	`11{x}`
[7]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
PMU	0xE08	PMIIDR	None

This interface is accessible as follows:

When DoubleLockStatus(), or !IsCorePowered() or !AllowExternalPMUAccess()

ERROR

When OSLockStatus()

ERROR

Otherwise

RO

B.6.9 PMCEID0, Performance Monitors Common Event Identification register 0

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range `0x0000` to `0x001F`.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

This view of the register was previously called PMCEID0_EL0.

Configurations

PMCEID0 is in the Core power domain.

External register PMCEID0 bits [31:0] are architecturally mapped to AArch64 System register A.8.3 PMCEID0_EL0, Performance Monitors Common Event Identification Register 0 on page 480 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xE20

Access type

RO

Reset value

```
0111 1111 1111 1111 0110 1111 0011 1111
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-109: PMU_PMCEID0 bit assignments

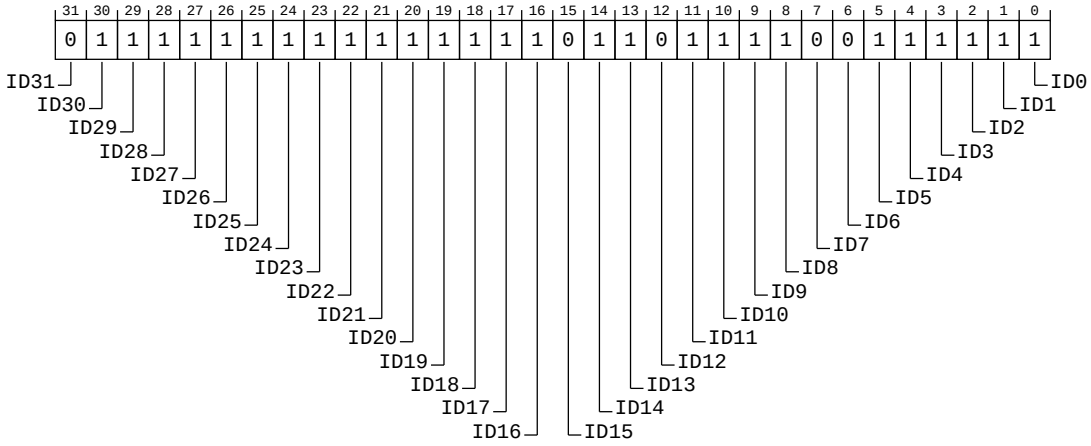


Table B-223: PMCEID0 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID[n] corresponds to Common event n. For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[30]	ID30	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[29]	ID29	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[28]	ID28	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[27]	ID27	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[26]	ID26	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[25]	ID25	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[24]	ID24	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[23]	ID23	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[22]	ID22	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[21]	ID21	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[20]	ID20	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[19]	ID19	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[18]	ID18	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[17]	ID17	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[16]	ID16	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[15]	ID15	ID[n] corresponds to Common event n. For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[14]	ID14	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[13]	ID13	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[12]	ID12	ID[n] corresponds to Common event n. For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[11]	ID11	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[10]	ID10	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[9]	ID9	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[8]	ID8	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[7]	ID7	ID[n] corresponds to Common event n. For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[6]	ID6	ID[n] corresponds to Common event n. For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[5]	ID5	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[4]	ID4	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[3]	ID3	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[2]	ID2	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[1]	ID1	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1
[0]	ID0	ID[n] corresponds to Common event n. For each bit: **0b1** The Common event is implemented.	0b1

Accessibility


Note

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE20	PMCEID0	None

This interface is accessible as follows:

When DoubleLockStatus(), or !IsCorePowered() or !AllowExternalPMUAccess()

ERROR

When OSLockStatus()

ERROR

Otherwise

RO

B.6.10 PMCEID1, Performance Monitors Common Event Identification register 1

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range `0x020` to `0x03F`.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.


Note

This view of the register was previously called PMCEID1_EL0.

Configurations

PMCEID1 is in the Core power domain.

External register PMCEID1 bits [31:0] are architecturally mapped to AArch64 System register A.8.4 PMCEID1_EL0, Performance Monitors Common Event Identification Register 1 on page 491 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xE24

Access type

RO

Reset value

```
1111 1110 1111 0010 1010 1110 0111 1111
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-110: PMU_PMCEID1 bit assignments

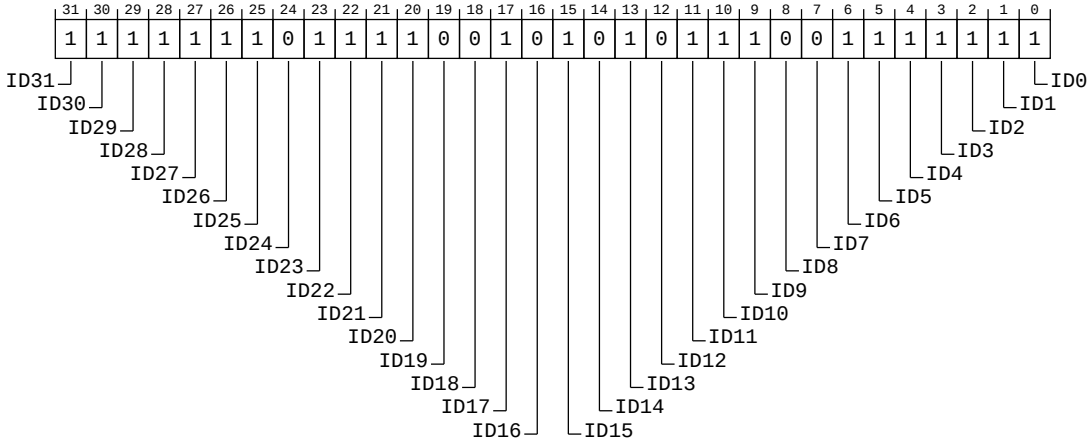


Table B-225: PMCEID1 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[30]	ID30	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[29]	ID29	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[28]	ID28	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[27]	ID27	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[26]	ID26	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[25]	ID25	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[24]	ID24	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[23]	ID23	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[22]	ID22	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[21]	ID21	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[20]	ID20	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[19]	ID19	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[18]	ID18	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[17]	ID17	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[16]	ID16	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[15]	ID15	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[14]	ID14	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[13]	ID13	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[12]	ID12	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[11]	ID11	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[10]	ID10	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[9]	ID9	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[8]	ID8	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[7]	ID7	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[6]	ID6	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[5]	ID5	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[4]	ID4	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[3]	ID3	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[2]	ID2	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[1]	ID1	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[0]	ID0	ID[n] corresponds to Common event (0x0020 + n). For each bit: **0b1** The Common event is implemented.	0b1

Accessibility


Note

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE24	PMCEID1	None

This interface is accessible as follows:

When DoubleLockStatus(), or !IsCorePowered() or !AllowExternalPMUAccess()
ERROR

When OSLockStatus()
ERROR

Otherwise
RO

B.6.11 PMCEID2, Performance Monitors Common Event Identification register 2

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x4000 to 0x401F.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

PMCEID2 is in the Core power domain.

External register PMCEID2 bits [31:0] are architecturally mapped to AArch64 System register A.8.3 PMCEID0_EL0, Performance Monitors Common Event Identification Register 0 on page 480 bits [63:32].

Attributes

Width

32

Component

PMU

Register offset

0xE28

Access type

RO

Reset value

```
xxxx 1111 xxxx 1111 0001 101x x111 1111
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-111: PMU_PMCEID2 bit assignments

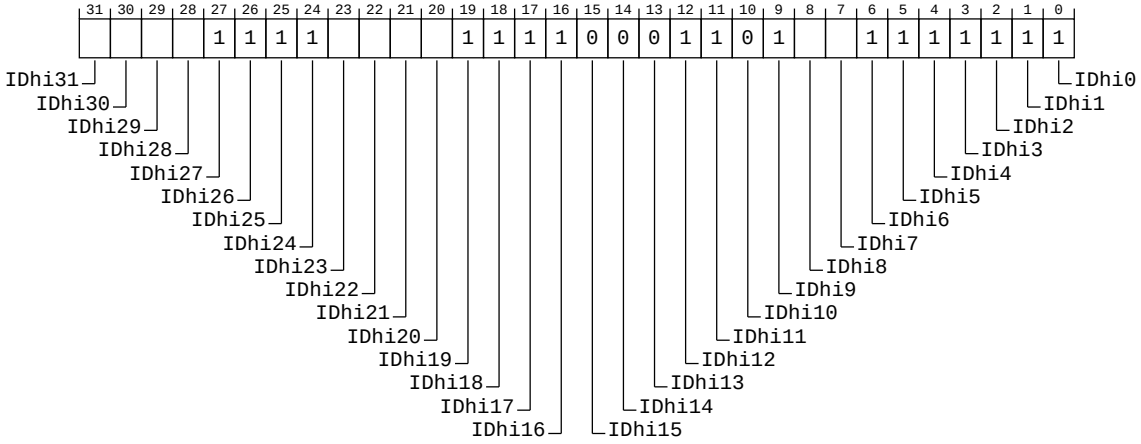


Table B-227: PMCEID2 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[30]	IDhi30	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[29]	IDhi29	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[28]	IDhi28	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[27]	IDhi27	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[26]	IDhi26	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[25]	IDhi25	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[24]	IDhi24	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[23]	IDhi23	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[22]	IDhi22	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[21]	IDhi21	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[20]	IDhi20	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.

Bits	Name	Description	Reset
[19]	IDhi19	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[18]	IDhi18	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[17]	IDhi17	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[16]	IDhi16	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[15]	IDhi15	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[14]	IDhi14	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[13]	IDhi13	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[12]	IDhi12	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[11]	IDhi11	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[10]	IDhi10	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[9]	IDhi9	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[8]	IDhi8	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[7]	IDhi7	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[6]	IDhi6	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[5]	IDhi5	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[4]	IDhi4	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[3]	IDhi3	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[2]	IDhi2	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[1]	IDhi1	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1
[0]	IDhi0	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: **0b1** The Common event is implemented.	0b1

Accessibility


Note

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE28	PMCEID2	None

This interface is accessible as follows:

When DoubleLockStatus(), or !IsCorePowered() or !AllowExternalPMUAccess()

ERROR

When OSLockStatus()

ERROR

Otherwise

RO

B.6.12 PMCEID3, Performance Monitors Common Event Identification register 3

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range `0x4020` to `0x403F`.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

PMCEID3 is in the Core power domain.

External register PMCEID3 bits [31:0] are architecturally mapped to AArch64 System register A.8.4 PMCEID1_EL0, Performance Monitors Common Event Identification Register 1 on page 491 bits [63:32].

Attributes

Width

32

Component

PMU

Register offset

0xE2C

Access type

RO

Reset value

```
0000 0000 xx00 x000 xxxx xxxx x111 x111
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-112: PMU_PMCEID3 bit assignments

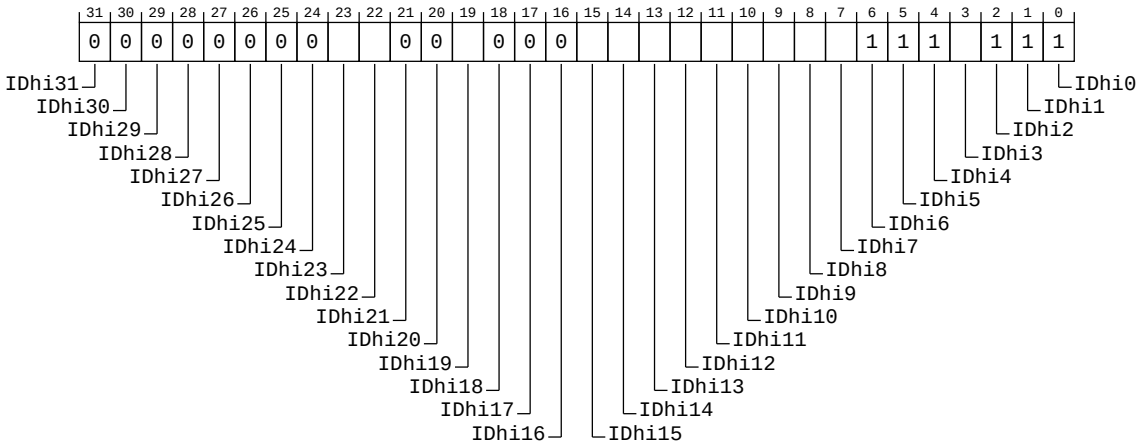


Table B-229: PMCEID3 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[30]	IDhi30	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[29]	IDhi29	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[28]	IDhi28	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[27]	IDhi27	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[26]	IDhi26	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[25]	IDhi25	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[24]	IDhi24	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[23]	IDhi23	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[22]	IDhi22	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[21]	IDhi21	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[20]	IDhi20	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[19]	IDhi19	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[18]	IDhi18	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[17]	IDhi17	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[16]	IDhi16	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted.	0b0
[15]	IDhi15	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.

Bits	Name	Description	Reset
[14]	IDhi14	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[13]	IDhi13	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[12]	IDhi12	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[11]	IDhi11	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[10]	IDhi10	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.

Bits	Name	Description	Reset
[9]	IDhi9	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[8]	IDhi8	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[7]	IDhi7	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[6]	IDhi6	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[5]	IDhi5	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[4]	IDhi4	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[3]	IDhi3	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b0** The Common event is not implemented, or not counted. **0b1** The Common event is implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[2]	IDhi2	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[1]	IDhi1	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1
[0]	IDhi0	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: **0b1** The Common event is implemented.	0b1

Accessibility


Note

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE2C	PMCEID3	None

This interface is accessible as follows:

When DoubleLockStatus(), or !IsCorePowered() or !AllowExternalPMUAccess()

ERROR

When OSLockStatus()

ERROR

Otherwise

RO

B.6.13 PMSSCR, PMU Snapshot Capture Register

Provides a mechanism for software to initiate a sample.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE30

Access type

WO

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-113: PMU_PMSSCR bit assignments

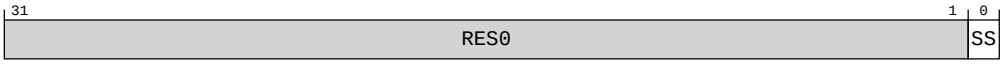


Table B-231: PMSSCR bit descriptions

Bits	Name	Description	Reset
[31:1]	**RES0**	Reserved	**RES0**
[0]	SS	Capture now. **0b0** Ignored. **0b1** Initiate a capture immediately.	x

Accessibility

Component	Offset	Instance	Range
PMU	0xE30	PMSSCR	None

This interface is accessible as follows:

WO

B.6.14 PMMIR, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation.

Configurations

PMMIR is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xE40

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 1010
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-114: PMU_PMMIR bit assignments

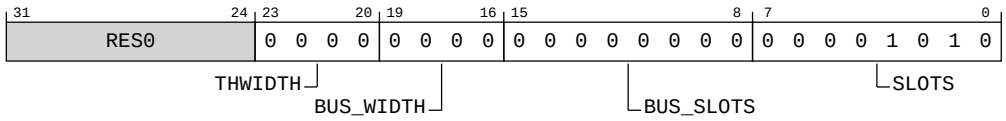


Table B-233: PMMIR bit descriptions

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[23:20]	THWIDTH	PMEVTYPER<n>_EL0.TH width. Indicates implementation of the FEAT_PMUv3_TH feature, and, if implemented, the size of the PMEVTYPER<n>_EL0.TH field. **0b0000** FEAT_PMUv3_TH is not implemented.	0b0000
[19:16]	BUS_WIDTH	Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as Log_2(number of bytes), plus one. **0b0000** The information is not available.	0b0000
[15:8]	BUS_SLOTS	Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle. **0x00** The largest value by which the BUS_ACCESS PMU event may increment in one cycle is 0.	0x00
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero. **0x0A** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 10.	0x0A

Accessibility

If the Core power domain is off or in a low-power state, access to this register returns an Error.

Component	Offset	Instance	Range
PMU	0xE40	PMMIR	31:0

This interface is accessible as follows:

When DoubleLockStatus(), or !IsCorePowered() or !AllowExternalPMUAccess()
ERROR

When OSLockStatus()
ERROR

Otherwise
RO

B.6.15 PMDEVARCH, Performance Monitors Device Architecture register

Identifies the programmers' model architecture of the Performance Monitor component.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width
32

Component

PMU

Register offset

0xFBC

Access type

RO

Reset value

```
0100 0111 0111 0000 0010 1010 0001 0110
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-115: PMU_PMDEVARCH bit assignments

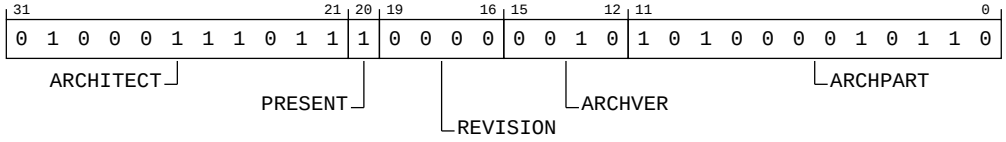


Table B-235: PMDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architect of the component. For Performance 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`
[20]	PRESENT	DEVARCH present. Indicates that the PMDEVARCH register is present. **0b1**	`0b1`
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. For Performance Monitors, the revision defined by Armv8 is `0x0`. All other values are reserved. **0b0000**	`0b0000`
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **0b0010** Performance Monitors Extension version 3, PMUv3.	`0b0010`
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **0xA16** Armv8-A PE performance monitors, including the 32-bit programmers' model extension.	`0xA16`

Accessibility

Component	Offset	Instance	Range
PMU	0xFBC	PMDEVARCH	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.6.16 PMDEVID, Performance Monitors Device ID register

Provides information about features of the Performance Monitors implementation.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required from Armv8.2 and in any implementation that includes FEAT_PCSRv8p2. Otherwise, its location is **RES0**.


Note

Before Armv8.2, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of EDDEVID.PCSample.

Attributes

Width

32

Component

PMU

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-116: PMU_PMDEVID bit assignments

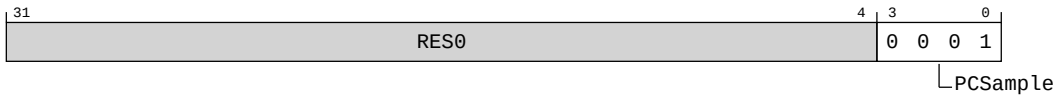


Table B-237: PMDEVID bit descriptions

Bits	Name	Description	Reset
[31:4]	**RES0**	Reserved	**RES0**
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using Performance Monitors registers. **0b0001** PC Sample-based Profiling Extension is implemented in the Performance Monitors register space.	0b0001

Accessibility

Component	Offset	Instance	Range
PMU	0xFC8	PMDEVID	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.6.17 PMDEVTYPE, Performance Monitors Device Type register

Indicates to a debugger that this component is part of a PE's performance monitor interface.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

PMU

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-117: PMU_PMDEVTYPE bit assignments

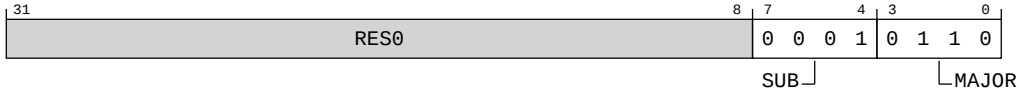


Table B-239: PMDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Subtype. Indicates this is a component within a PE. **0b0001**	0b0001
[3:0]	MAJOR	Major type. Indicates this is a performance monitor component. **0b0110**	0b0110

Accessibility

Component	Offset	Instance	Range
PMU	0xFCC	PMDEVTYPE	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.6.18 PMPIDR4, Performance Monitors Peripheral Identification Register 4

Provides information to identify a Performance Monitor component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFD0

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-118: PMU_PMPIDR4 bit assignments

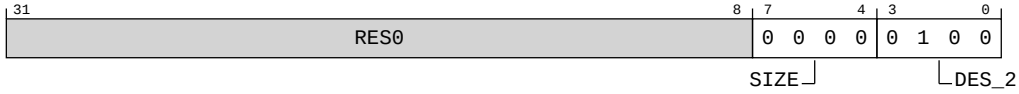


Table B-241: PMPIDR4 bit descriptions

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

Accessibility

Component	Offset	Instance	Range
PMU	0xFD0	PMPIDR4	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.6.19 PMPIDR0, Performance Monitors Peripheral Identification Register 0

Provides information to identify a Performance Monitor component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1000 1100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-119: PMU_PMPIDR0 bit assignments

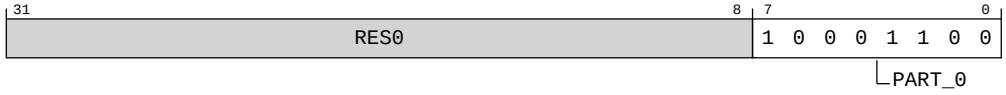


Table B-243: PMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	PART_0	Part number, least significant byte. **0x8C** C1-Ultra	0x8C

Accessibility

Component	Offset	Instance	Range
PMU	0xFE0	PMPIDR0	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.6.20 PMPIDR1, Performance Monitors Peripheral Identification Register 1

Provides information to identify a Performance Monitor component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE4

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-120: PMU_PMPIDR1 bit assignments

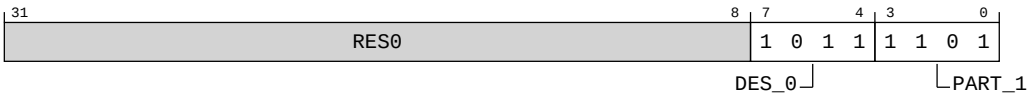


Table B-245: PMPIDR1 bit descriptions

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-Ultra	`0b1101`

Accessibility

Component	Offset	Instance	Range
PMU	0xFE4	PMPIDR1	None

This interface is accessible as follows:

When !IsCorePowered()
ERROR

Otherwise
RO

B.6.21 PMPIDR2, Performance Monitors Peripheral Identification Register 2

Provides information to identify a Performance Monitor component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE8

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-121: PMU_PMPIDR2 bit assignments

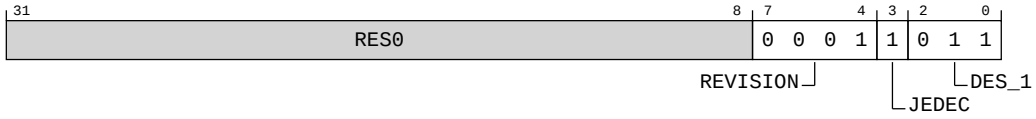


Table B-247: PMPIDR2 bit descriptions

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** r1p0	0b0001
[3]	JEDEC	Indicates a JEP106 identity code is used. **0b1**	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	Instance	Range
PMU	0xFE8	PMPIDR2	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.6.22 PMPIDR3, Performance Monitors Peripheral Identification Register 3

Provides information to identify a Performance Monitor component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFEC

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-122: PMU_PMPIDR3 bit assignments

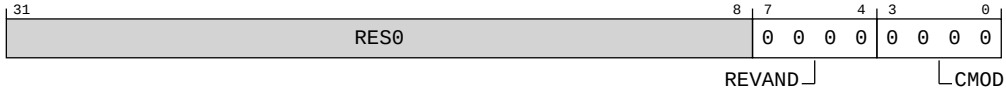


Table B-249: PMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7:4]	REVAND	Part minor revision. Parts using PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0000** r1p0	0b0000
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
PMU	0xFEC	PMPIDR3	None

This interface is accessible as follows:

When !IsCorePowered()
ERROR

Otherwise
RO

B.6.23 PMCIDR0, Performance Monitors Component Identification Register 0

Provides information to identify a Performance Monitor component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width
32

Component
PMU

Register offset
0xFF0

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-123: PMU_PMCIDR0 bit assignments

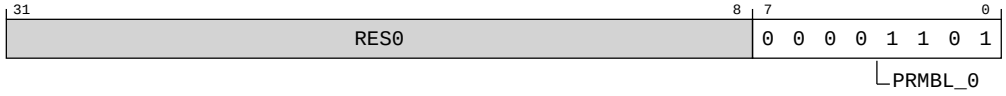


Table B-251: PMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble. **0x0D**	0x0D

Accessibility

Component	Offset	Instance	Range
PMU	0xFF0	PMCIDR0	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.6.24 PMCIDR1, Performance Monitors Component Identification Register 1

Provides information to identify a Performance Monitor component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFF4

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-124: PMU_PMCIDR1 bit assignments

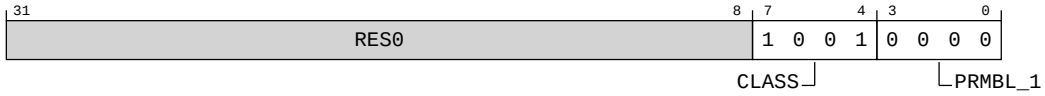


Table B-253: PMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight component.	0b1001
[3:0]	PRMBL_1	Preamble. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
PMU	0xFF4	PMCIDR1	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.6.25 PMCIDR2, Performance Monitors Component Identification Register 2

Provides information to identify a Performance Monitor component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFF8

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-125: PMU_PMCIDR2 bit assignments

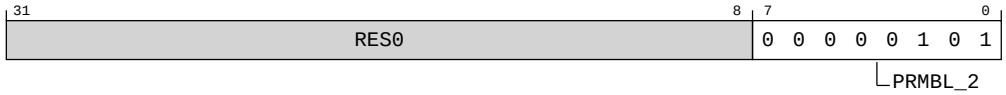


Table B-255: PMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble. **0x05**	0x05

Accessibility

Component	Offset	Instance	Range
PMU	0xFF8	PMCIDR2	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.6.26 PMCIDR3, Performance Monitors Component Identification Register 3

Provides information to identify a Performance Monitor component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFFC

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-126: PMU_PMCIDR3 bit assignments

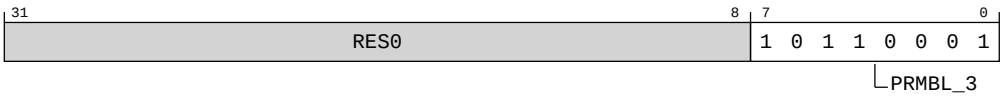


Table B-257: PMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble. **0xB1**	0xB1

Accessibility

Component	Offset	Instance	Range
PMU	0xFFC	PMCIDR3	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.7 External PPM registers summary

The following summary table provides an overview of all memory-mapped PPM 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-259: PPM registers summary

Offset	Name	Reset	Width	Description
0x000	CPUPPMCR	See individual bit resets.	64-bit	Global Performance and Power Management Configuration Register
0x010	CPUMPMMCR	See individual bit resets.	64-bit	Global MPMM Control Register
0x020	CPUPPMPDPCR	See individual bit resets.	64-bit	Performance and Power Management PDP Control Register
0x080	CPUPPMCR4	See individual bit resets.	64-bit	Power Performance Management Register
0x088	CPUPPMCR5	See individual bit resets.	64-bit	Power Performance Management Register

Offset	Name	Reset	Width	Description
0x090	CPUPPMCR6	See individual bit resets.	64-bit	Power Performance Management Register
0x0A0	CPUACTMCTL	See individual bit resets.	64-bit	CPU Power Performance Management Control Register

B.7.1 CPUPPMCR, Global Performance and Power Management Configuration Register

Provides **IMPLEMENTATION DEFINED** control and discovery of the Performance and Power Management (PPM) features.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x000

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxx0 xxxx xx00 0000 xxxx xxxx xxxx x111 xxxx x011 xxxx xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-127: EXT_CPUPPMCR bit assignments

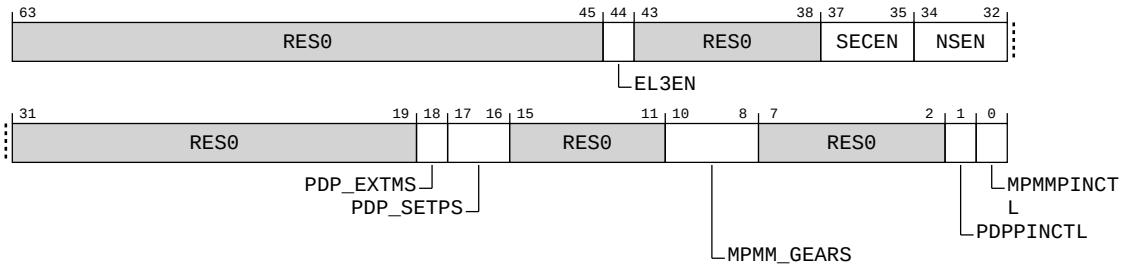


Table B-260: CPUPPMCR bit descriptions

Bits	Name	Description	Reset
[63:45]	**RES0**	Reserved	**RES0**
[44]	EL3EN	EL3 Activity Meter Count Enable. Enables Activity Meter update in EL3 - Root or EL3 CPU Activity is counted if this bit is set **0b0** Inhibit update of Activity Meter when core is in EL3 **0b1** Allow update of Activity Meter when core is in EL3	0b0
[43:38]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[37:35]	SECEN	Secure ELx Activity Meter Count Enable. Enables Activity Meter update in Secure ELx. - Each bit if set will enable counting in that particular secure EL **0b000** Inhibit update of Activity Meter when core is in Secure EL0/EL1/EL2 **0b001** Inhibit update of Activity Meter when core is in Secure EL1/EL2, and allow update in Secure EL0 **0b010** Inhibit update of Activity Meter when core is in Secure EL0/EL2, and allow update in Secure EL1 **0b011** Inhibit update of Activity Meter when core is in Secure EL2, and allow update in Secure EL0/EL1 **0b100** Inhibit update of Activity Meter when core is in Secure EL0/EL1, and allow update in Secure EL2 **0b101** Inhibit update of Activity Meter when core is in Secure EL1, and allow update in Secure EL0/EL2 **0b110** Inhibit update of Activity Meter when core is in Secure EL0, and allow update in Secure EL1/EL2 **0b111** Allow update of Activity Meter when core is in Secure EL0/EL1/EL2	0b000

Bits	Name	Description	Reset
[34:32]	NSEN	Non-secure ELx Activity Meter Count Enable. Enables Activity Meter update in Non-secure ELx. - Each bit if set will enable counting in that particular non-secure EL **0b000** Inhibit update of Activity Meter when core is in Non-secure EL0/EL1/EL2 **0b001** Inhibit update of Activity Meter when core is in Non-secure EL1/EL2, and allow update in Non-secure EL0 **0b010** Inhibit update of Activity Meter when core is in Non-secure EL0/EL2, and allow update in Non-secure EL1 **0b011** Inhibit update of Activity Meter when core is in Non-secure EL2, and allow update in Non-secure EL0/EL1 **0b100** Inhibit update of Activity Meter when core is in Non-secure EL0/EL1, and allow update in Non-secure EL2 **0b101** Inhibit update of Activity Meter when core is in Non-secure EL1, and allow update in Non-secure EL0/EL2 **0b110** Inhibit update of Activity Meter when core is in Non-secure EL0, and allow update in Non-secure EL1/EL2 **0b111** Allow update of Activity Meter when core is in Non-secure EL0/EL1/EL2	0b000
[31:19]	**RES0**	Reserved	**RES0**
[18]	PDP_EXTMS	External memory system PDP control **0b0** Independent external memory system PDP control is not implemented **0b1** Independent external memory system PDP control is implemented	0b1
[17:16]	PDP_SETPS	Number of PDP Setpoints Implemented **0b11** 3 PDP setpoints are implemented	0b11
[15:11]	**RES0**	Reserved	**RES0**
[10:8]	MPMM_GEARS	Number of MPMM Gears Implemented **0b011** 3 MPMM gears are implemented	0b011
[7:2]	**RES0**	Reserved	**RES0**
[1]	PDPPINCTL	PDP Pin Control Enabled **0b0** PDP control through SPR and utility bus **0b1** PDP control through pin only	0b0

Bits	Name	Description	Reset
[0]	MPMMPINCTL	MPMM Pin Control Enabled **0b0** MPMM control through SPR and utility bus **0b1** MPMM control through pin only	0b0

Accessibility

Component	Offset	Range
PPM	0x000	None

This interface is accessible as follows:

RW

B.7.2 CPUMPMMCR, Global MPMM Control Register

Provides **IMPLEMENTATION DEFINED** control of the Maximum Power Mitigation Mechanism (MPMM) feature.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x010

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 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-128: EXT_CPUMPMMCR bit assignments

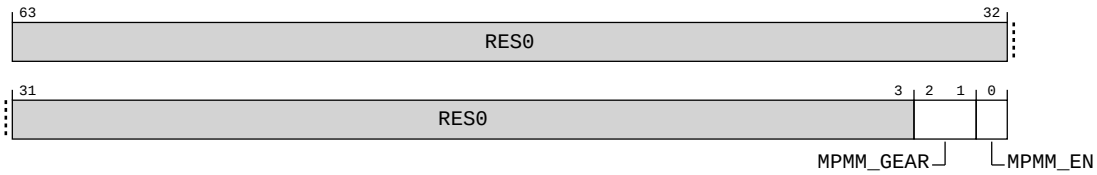


Table B-262: CPUMPMMCR bit descriptions

Bits	Name	Description	Reset
[63:3]	**RES0**	Reserved	**RES0**
[2:1]	MPMM_GEAR	MPMM Gear Select **0b00** Select MPMM Gear 0 **0b01** Select MPMM Gear 1 **0b10** Select MPMM Gear 2 **0b11** Select MPMM Gear 3	0b00
[0]	MPMM_EN	MPMM Global Enable **0b0** MPMM is disabled **0b1** MPMM is enabled	0b0

Accessibility

Component	Offset	Range
PPM	0x010	None

This interface is accessible as follows:

RW

B.7.3 CPUPPMPDPCR, Performance and Power Management PDP Control Register

Provides **IMPLEMENTATION DEFINED** control of the Performance Defined Power (PDP) feature.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x020

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-129: EXT_CPUPPMPDPCR bit assignments

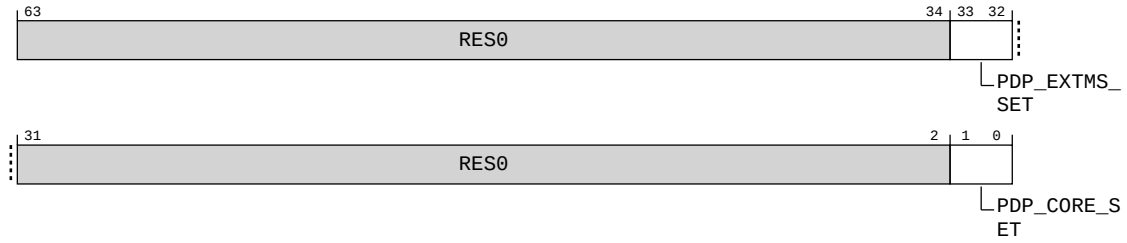


Table B-264: CPUPPMPDPCR bit descriptions

Bits	Name	Description	Reset
[63:34]	**RES0**	Reserved	**RES0**
[33:32]	PDP_EXTMS_SET	External memory system PDP Aggressiveness **0b00** Disable PDP **0b01** Enable PDP at low aggressiveness **0b10** Enable PDP at medium aggressiveness **0b11** Enable PDP at high aggressiveness	0b00

Bits	Name	Description	Reset
[31:2]	**RES0**	Reserved	**RES0**
[1:0]	PDP_CORE_SET	Core PDP Aggressiveness **0b00** Disable PDP **0b01** Enable PDP at low aggressiveness **0b10** Enable PDP at medium aggressiveness **0b11** Enable PDP at high aggressiveness	0b00

Accessibility

Component	Offset	Range
PPM	0x020	None

This interface is accessible as follows:

RW

B.7.4 CPUPPMCR4, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x080

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-130: EXT_CPUPPMCR4 bit assignments

Table B-266: CPUPPMCR4 bit descriptions

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

Accessibility

Component	Offset	Range
PPM	0x080	None

This interface is accessible as follows:

RW

B.7.5 CPUPPMCR5, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x088

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

Reserved

Figure B-131: EXT_CPUPPMCR5 bit assignments

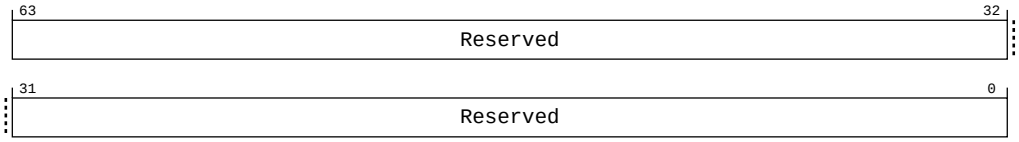


Table B-268: CPUPPMCR5 bit descriptions

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

Accessibility

Component	Offset	Range
PPM	0x088	None

This interface is accessible as follows:

RW

B.7.6 CPUPPMCR6, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x090

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

Reserved

Figure B-132: EXT_CPUPPMCR6 bit assignments

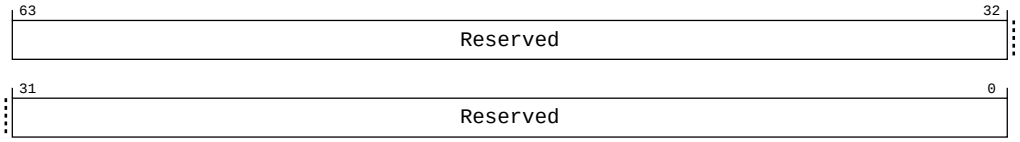


Table B-270: CPUPPMCR6 bit descriptions

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

Accessibility

Component	Offset	Range
PPM	0x090	None

This interface is accessible as follows:

RW

B.7.7 CPUACTMCTL, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x0A0

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-133: EXT_CPUACTMCTL bit assignments

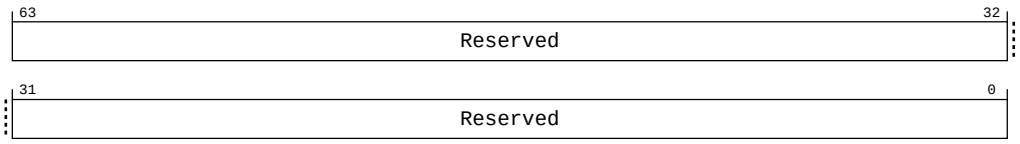


Table B-272: CPUACTMCTL bit descriptions

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

Accessibility

Component	Offset	Range
PPM	0x0A0	None

This interface is accessible as follows:

RW

B.8 External RAS registers summary

The following summary table provides an overview of all 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.

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-274: 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	0x0000000040000062	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	0xD8C1083B	32-bit	Implementation Identification Register
0xFA8	ERRDEVAFF	See individual bit resets.	64-bit	Device Affinity Register
0xFBC	ERRDEVARCH	See individual bit resets.	32-bit	Device Architecture Register
0xFC8	ERRDEVID	0x00000001	32-bit	Device Configuration Register
0xFD0	ERRPIDR4	0x00000004	32-bit	Peripheral Identification Register 4
0xFE0	ERRPIDR0	0x0000008C	32-bit	Peripheral Identification Register 0
0xFE4	ERRPIDR1	0x000000BD	32-bit	Peripheral Identification Register 1
0xFE8	ERRPIDR2	See individual bit resets.	32-bit	Peripheral Identification Register 2
0xFEC	ERRPIDR3	See individual bit resets.	32-bit	Peripheral Identification Register 3

Offset	Name	Reset	Width	Description
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.8.1 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.

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-134: EXT_ERR0FR bit assignments

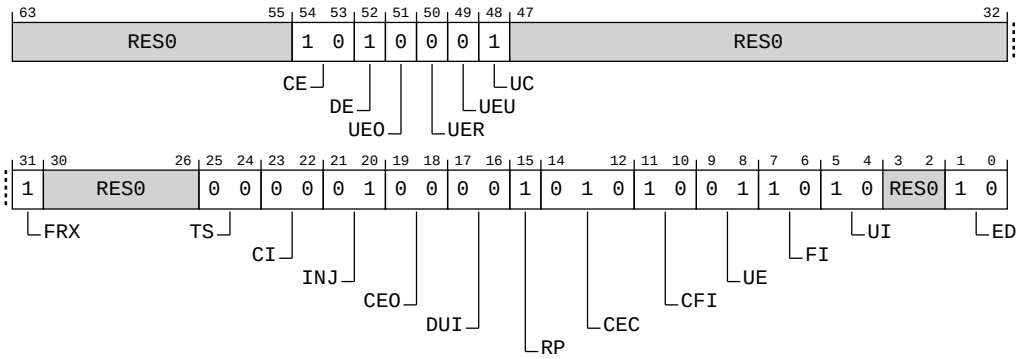


Table B-275: ERR0FR bit descriptions

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 ERXSTATUS_EL1.CE to 0b10.	0b10
[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	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. **0b0** Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors.	0b0
[48]	UC	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. **0b1** Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors.	0b1
[47:32]	**RES0**	Reserved	**RES0**
[31]	FRX	Feature Register extension. Defines whether ERXFR_EL1[63:48] are architecturally defined. **0b1** ERXFR_EL1[63:48] are defined by the architecture.	0b1
[30:26]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[25:24]	TS	Timestamp Extension. Indicates whether, for each error record <m> owned by this node, ERXMISC3_EL1 is used as the timestamp register, and, if it is, the timebase used by the timestamp. **0b00** The node does not support a timestamp register.	0b00
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented. **0b00** Does not support the critical error interrupt. ERXCTLR_EL1.CI is **RES0**.	0b00
[21:20]	INJ	Fault Injection Extension. Indicates whether the RAS Common Fault Injection Model Extension is implemented. **0b01** The node implements the RAS Common Fault Injection Model Extension. See ERXPFGF_EL1 for more information.	0b01
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior when a second Corrected error is detected after a first Corrected error has been recorded by an error record <m> owned by the node. **0b00** Counts Corrected errors if a counter is implemented. Keeps the previous error syndrome. If the counter overflows, or no counter is implemented, then ERXSTATUS_EL1.OF is set to 0b1.	0b00
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the control for enabling error recovery interrupts on deferred errors are implemented. **0b00** Does not support the control for enabling error recovery interrupts on deferred errors. ERXCTLR_EL1.DUI is **RES0**.	0b00
[15]	RP	Repeat counter. Indicates whether the node implements the repeat Corrected error counter in ERXMISC0_EL1 for each error record <m> owned by the node that implements the standard Corrected error counter. **0b1** A first (repeat) counter and a second (other) counter are implemented. The repeat counter is the same size as the primary error counter.	0b1
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter (CE counter) mechanisms in ERXMISC0_EL1 for each error record <m> owned by the node that can record countable errors. **0b010** Implements an 8-bit Corrected error counter in ERXMISC0_EL1[39:32].	0b010
[11:10]	CFI	Fault handling interrupt for corrected errors. Indicates whether the control for enabling fault handling interrupts on corrected errors are implemented. **0b10** Control for enabling fault handling interrupts on corrected errors is supported and controllable using ERXCTLR_EL1.CFI.	0b10
[9:8]	UE	In-band uncorrected error reporting. Indicates whether the in-band uncorrected error reporting (External Aborts) and associated controls are implemented. **0b01** In-band uncorrected error reporting (External Aborts) is supported and always enabled. ERXCTLR_EL1.UE is **RES0**.	0b01
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented. **0b10** Fault handling interrupt is supported and controllable using ERXCTLR_EL1.FI.	0b10

Bits	Name	Description	Reset
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented. **0b10** Error handling interrupt is supported and controllable using ERXCTLR_EL1.UI.	0b10
[3:2]	**RES0**	Reserved	**RES0**
[1:0]	ED	Error reporting and logging. Indicates whether error record <n> is the first record owned the node, and, if so, whether it implements the controls for enabling and disabling error reporting and logging. **0b10** Error reporting and logging is controllable using ERXCTLR_EL1.ED.	0b10

Accessibility

Component	Offset	Instance	Range
RAS	0x0	ERR0FR	None

This interface is accessible as follows:

RO

B.8.2 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**.

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-135: EXT_ERR0CTLR bit assignments

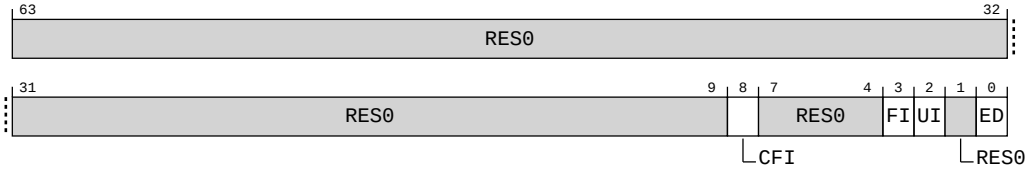


Table B-277: ERR0CTLR bit descriptions

Bits	Name	Description	Reset
[63:9]	**RES0**	Reserved	**RES0**
[8]	CFI	Fault handling interrupt for Corrected errors enable. This control applies to errors arising from both reads and writes. The fault handling interrupt is generated when one of the standard CE counters on ERXMISC0_EL1 overflows and the overflow bit is set. The possible values are: **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. Cold reset only. Unaffected by Warm reset	0b0
[7:4]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[3]	FI	Fault handling interrupt enable. This control applies to errors arising from both reads and writes. The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. The possible values are: **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. Cold reset only. Unaffected by Warm reset	0b0
[2]	UI	Uncorrected error recovery interrupt enable. This control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred. **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. Cold reset only. Unaffected by Warm reset	0b0
[1]	**RES0**	Reserved	**RES0**
[0]	ED	Error Detection and correction enable. The possible values are: **0b0** Error detection and correction disabled. **0b1** Error detection and correction enabled. Cold reset only. Unaffected by Warm reset	0b0

Accessibility

Component	Offset	Instance	Range
RAS	0x8	ERR0CTLR	None

This interface is accessible as follows:

RW

B.8.3 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.

Configurations

ERR<n>FR describes the features implemented by the node.

Attributes

Width

64

Component

RAS

Register offset

0x10

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0000 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.

Bit descriptions

Figure B-136: EXT_ERR0STATUS bit assignments

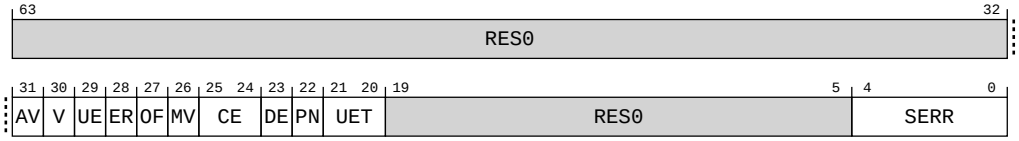


Table B-279: ERR0STATUS bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	AV	Address Valid. The possible values are: **0b0** ERXADDR_EL1 not valid. **0b1** ERXADDR_EL1 contains an address associated with the highest priority error recorded by this record. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0
[30]	V	Status Register Valid. The possible values are: **0b0** ERXSTATUS_EL1 not valid. **0b1** ERXSTATUS_EL1 valid. At least one error has been recorded. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0

Bits	Name	Description	Reset
[29]	UE	Uncorrected Error. The possible values are: **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 ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0
[28]	ER	Error Reported. The possible values are: **0b0** No in-band error (External Abort) reported. **0b1** An External Abort was signaled by the node to the requester making the access or other transaction. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset **Note:** An External Abort signaled by the node might be masked and not generate any exception.	0b0
[27]	OF	Overflow. The possible values are: **0b0** If UE == 1, then no error status for an Uncorrected error has been discarded. If UE == 0 and DE == 1, then no error status for a Deferred error has been discarded. If UE == 0, DE == 0, and CE !== 0b00, then the corrected error counter has not overflowed. **0b1** More than one error has occurred and so details of the other error have been discarded. When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. Cold reset only. Unaffected by Warm reset This bit is read/write-one-to-clear.	0b0

Bits	Name	Description	Reset
[26]	MV	Miscellaneous Registers Valid. The possible values are: **0b0** ERXMISC<m>_EL1 not valid. **0b1** This bit indicates that the ERXMISC<m>_EL1 registers contain additional information for an error recorded by this record. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset **Note:** If the ERXMISC<m>_EL1 registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded.	0b0
[25:24]	CE	Corrected Error. The possible values are: **0b00** No errors were corrected. **0b01** At least one transient error was corrected. **0b10** At least one error was corrected. **0b11** At least one persistent error was corrected. When clearing ERXSTATUS_EL1.V to 0b0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. This field is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. This field is read/write-ones-to-clear. Writing a value other than all-zeros or all-ones sets this field to an **UNKNOWN** value. Cold reset only. Unaffected by Warm reset	0b00
[23]	DE	Deferred Error. The possible values are: **0b0** No errors were deferred. **0b1** At least one error was not corrected and deferred. When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0

Bits	Name	Description	Reset
[22]	PN	Poison. The value is: **0b0** This core cannot distinguish a poisoned value from a corrupted value. When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if any of the following are true: • ERXSTATUS_EL1.V == 0b0. • ERXSTATUS_EL1.{DE,UE} == {0,0}. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0
[21:20]	UET	Uncorrected Error Type. The value is: **0b00** Uncorrected error, Uncontainable error (UC). Cold reset only. Unaffected by Warm reset	0b00
[19:5]	**RES0**	Reserved	**RES0**
[4:0]	SERR	Primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry. The possible values are: **0b00000** No error **0b00010** ECC error from internal data buffer. **0b00110** ECC error on cache data RAM. **0b00111** ECC error on cache tag or dirty RAM. **0b01000** Parity error on TLB data RAM. **0b10010** Error response for a cache copyback. **0b10101** Deferred error from completer not supported at the consumer. For example, poisoned data received from a completer by a requester that cannot defer the error further. Cold reset only. Unaffected by Warm reset	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.

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:

RW

B.8.4 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. It is **IMPLEMENTATION DEFINED** how the recorded address maps to the software-visible physical address. Software might have to reconstruct the actual physical addresses using the identity of the node and knowledge of the system.

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
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-137: EXT_ERR0ADDR bit assignments

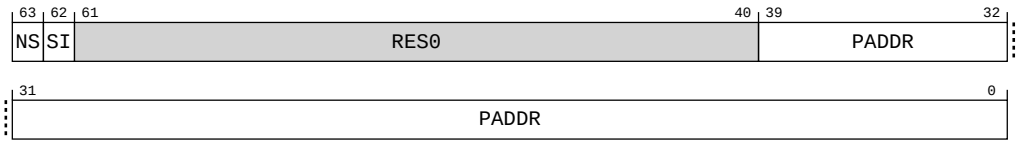


Table B-281: ERR0ADDR bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure attribute. **0b0** ERR<n>ADDR.PADDR is a Secure address. **0b1** ERR<n>ADDR.PADDR is a Non-secure address.	x
[62]	SI	Secure Incorrect. Indicates whether ERR<n>ADDR.NS is valid. **0b0** ERR<n>ADDR.NS is correct. That is, it matches the programmers' view of the Non-secure attribute for the recorded location. **0b1** ERR<n>ADDR.NS might not be correct, and might not match the programmers' view of the Non-secure attribute for the recorded location.	x
[61:40]	**RES0**	Reserved	**RES0**
[39:0]	PADDR	Physical Address [39:0]. Address of the recorded location	40{x}

Accessibility

Component	Offset	Instance	Range
RAS	0x18	ERR0ADDR	None

This interface is accessible as follows:

When ERRPFGF[FirstRecordOfNode(0)].AV == '0' and ext-ERR0STATUS.AV == '1'
RO

Otherwise
RW

B.8.5 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

Register offset

0x20

Access type

RW

Reset value

When ERRFR[FirstRecordOfNode(0)].CEC == '000'

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

When ERRFR[FirstRecordOfNode(0)].CEC == '100' and ERRFR[FirstRecordOfNode(0)].RP == '0'

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When ERRFR[FirstRecordOfNode(0)].CEC == '010' and ERRFR[FirstRecordOfNode(0)].RP == '0'

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When ERRFR[FirstRecordOfNode(0)].CEC == '100' and ERRFR[FirstRecordOfNode(0)].RP == '1'

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When ERRFR[FirstRecordOfNode(0)].CEC == '010' and ERRFR[FirstRecordOfNode(0)].RP == '1'

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 ERRFR[FirstRecordOfNode(0)].CEC == '000'

Figure B-138: EXT_ERR0MISC0 bit assignments

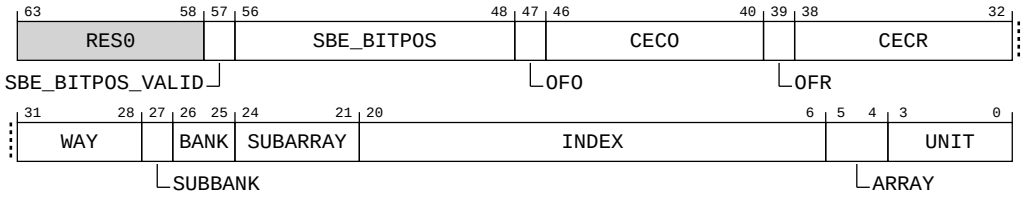


Table B-283: ERR0MISC0 bit descriptions

Bits	Name	Description	Reset
[63:58]	RES0	Reserved	RES0
[57]	SBE_BITPOS_VALID	Single Bit Error (SBE) bit position field ERXMISC0_EL1.SBE_BITPOS contains valid data **0b0** ERXMISC0_EL1.SBE_BITPOS does not contain valid data. **0b1** ERXMISC0_EL1.SBE_BITPOS contains valid data.	x
[56:48]	SBE_BITPOS	Single Bit Error (SBE) bit position. For a correctable error in a RAM with ECC (L1 data cache, L2 cache), indicates the bit position of the corrected error. Valid when ERXMISC0_EL1.SBE_BITPOS_VALID is 1'b1	9{x}

Bits	Name	Description	Reset
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERXMISC0_EL1.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 ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Cold or Warm reset.	x
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERXMISC0_EL1.CECR. Unaffected by Cold or Warm reset.	7{x}
[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERXMISC0_EL1.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 ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Cold or Warm reset.	x
[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. This field resets to an **IMPLEMENTATION DEFINED** which might be **UNKNOWN** on a Cold reset. If the reset value is **UNKNOWN**, then the value of this field remains **UNKNOWN** until software initializes it. Unaffected by Cold or Warm reset.	7{x}

Bits	Name	Description	Reset
[31:28]	WAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L1 Data Cache] • Indicates which Tag RAM way or data RAM way detected the error. Upper 2 bits are unused. [L2 TLB] • Indicates which RAM detected an error. The possible values are 0 (RAM 1) to 9 (RAM 10). [L1 Instruction Cache] • Indicates which way detected the error. Upper 2 bits are unused. [L2 Cache] • Indicates which way detected the error. Unaffected by Cold or Warm reset.	xxxx
[27]	SUBBANK	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L1 Instruction Cache] • Indicates which subbank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero. Unaffected by Cold or Warm reset.	x
[26:25]	BANK	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which L2 bank detected the error. [L1 Instruction Cache] • Indicates which bank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero. Unaffected by Cold or Warm reset.	xx
[24:21]	SUBARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which L2 data ECC granule detected the error. Upper bits are unused dependent on the ECC granule size. [L1 Data Cache] • Indicates for L1 Data RAM which word had the error detected. For L1 Tag RAMs which bank had the error (0b0000: bank0 , 0b0001: bank1) Unaffected by Cold or Warm reset.	xxxx

Bits	Name	Description	Reset
[20:6]	INDEX	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. [L1 Data Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size [L2 TLB] • Index of TLB RAM. Upper 4 bits are unused. [L1 Instruction Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. Unaffected by Cold or Warm reset.	`15{x}`

Bits	Name	Description	Reset
[5:4]	ARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] Indicates which array detected the error. The possible values are: • `0b00` L2 Tag RAM. • `0b01` L2 Data RAM. • `0b10` L2 TQ Data RAM. • `0b11` CHI Error. [L1 Data Cache] Indicates which array detected the error. The possible values are: • 00 LS Tag RAM 0. • 01 LS Tag RAM 1. • 10 LS Data RAM. • 11 LS Tag RAM 2. [L2 TLB] Indicates which array detected the error. The possible values are: • 00 Translation cache. • 01 GPT cache (when LEGACY_TZ_EN is 0). • 10 Reserved. • 11 Reserved. [L1 Instruction Cache] Indicates which array that detected the error, Data Array has higher priority. The possible values are: • `0b00` Tag. • `0b01` Data. Unaffected by Cold or Warm reset.	`xx`
[3:0]	UNIT	Indicates the unit which detected the error. The possible values are: **`0b0001`** L1 Instruction Cache. **`0b0010`** L2 TLB. **`0b0100`** L1 Data Cache. **`0b1000`** L2 Cache.	`xxxx`

When ERRFR[FirstRecordOfNode(0)].CEC == '100' and ERRFR[FirstRecordOfNode(0)].RP == '0'

Figure B-139: EXT_ERR0MISC0 bit assignments

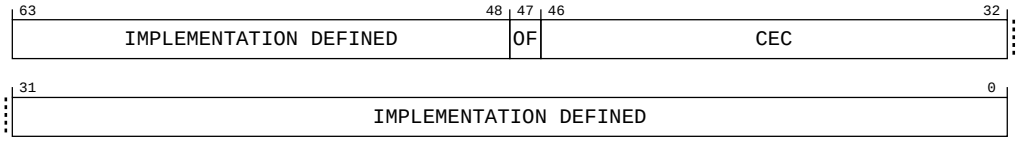


Table B-284: ERR0MISC0 bit descriptions

Bits	Name	Description	Reset
[63:48]	None	**IMPLEMENTATION DEFINED** syndrome.	16{x}
[47]	OF	Sticky overflow bit. Set to 1 when ERR0MISC0.CEC is incremented and wraps through zero. **0b0** Counter has not overflowed. **0b1** Counter has overflowed.	x
[46:32]	CEC	Corrected error count. Incremented for each Corrected error. It is **IMPLEMENTATION DEFINED** and might be **UNPREDICTABLE** whether Deferred and Uncorrected errors are counted.	15{x}
[31:0]	None	**IMPLEMENTATION DEFINED** syndrome.	32{x}

When ERRFR[FirstRecordOfNode(0)].CEC == '010' and ERRFR[FirstRecordOfNode(0)].RP == '0'

Figure B-140: EXT_ERR0MISC0 bit assignments

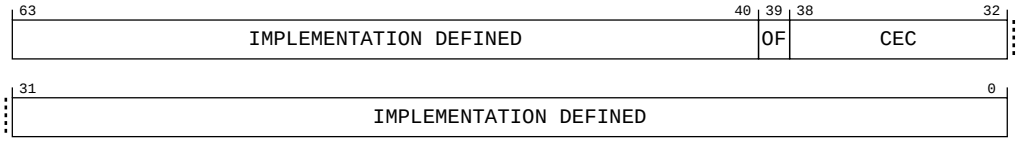


Table B-285: ERR0MISC0 bit descriptions

Bits	Name	Description	Reset
[63:40]	None	**IMPLEMENTATION DEFINED** syndrome.	24{x}
[39]	OF	Sticky overflow bit. Set to 1 when ERR0MISC0.CEC is incremented and wraps through zero. **0b0** Counter has not overflowed. **0b1** Counter has overflowed.	x
[38:32]	CEC	Corrected error count. Incremented for each Corrected error. It is **IMPLEMENTATION DEFINED** and might be **UNPREDICTABLE** whether Deferred and Uncorrected errors are counted.	7{x}
[31:0]	None	**IMPLEMENTATION DEFINED** syndrome.	32{x}

When ERRFR[FirstRecordOfNode(0)].CEC == '100' and ERRFR[FirstRecordOfNode(0)].RP == '1'

Figure B-141: EXT_ERR0MISC0 bit assignments

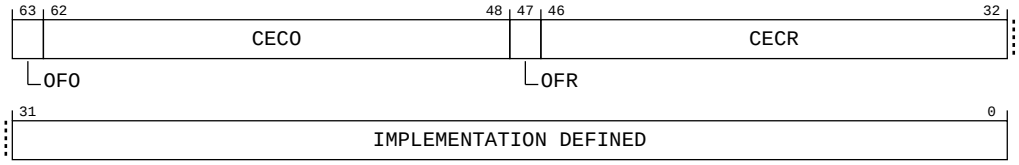


Table B-286: ERR0MISC0 bit descriptions

Bits	Name	Description	Reset
[63]	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.	x
[62:48]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERR0MISC0.CECR.	15{x}
[47]	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.	x
[46:32]	CECR	Corrected error count, repeat. Incremented for the first countable error, which also records other syndrome for the error, and subsequently for each countable error that matches the recorded other syndrome. Corrected errors are countable errors. It is **IMPLEMENTATION DEFINED** and might be **UNPREDICTABLE** whether Deferred and Uncorrected errors are countable errors.	15{x}
[31:0]	None	**IMPLEMENTATION DEFINED** syndrome.	32{x}

When ERRFR[FirstRecordOfNode(0)].CEC == '010' and ERRFR[FirstRecordOfNode(0)].RP == '1'

Figure B-142: EXT_ERR0MISC0 bit assignments

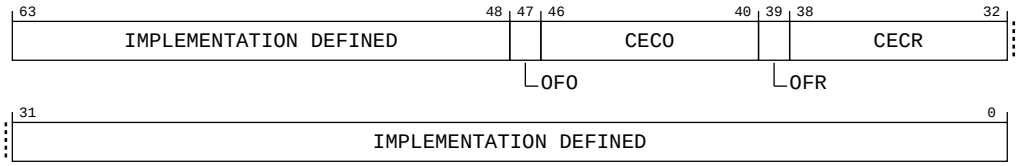


Table B-287: ERR0MISC0 bit descriptions

Bits	Name	Description	Reset
[63:48]	None	**IMPLEMENTATION DEFINED** syndrome.	16{x}

Bits	Name	Description	Reset
[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.	x
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERR0MISC0.CECR.	7{x}
[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.	x
[38:32]	CECR	Corrected error count, repeat. Incremented for the first countable error, which also records other syndrome for the error, and subsequently for each countable error that matches the recorded other syndrome. Corrected errors are countable errors. It is **IMPLEMENTATION DEFINED** and might be **UNPREDICTABLE** whether Deferred and Uncorrected errors are countable errors.	7{x}
[31:0]	None	**IMPLEMENTATION DEFINED** syndrome.	32{x}

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:

RW

B.8.6 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
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-143: EXT_ERR0MISC1 bit assignments

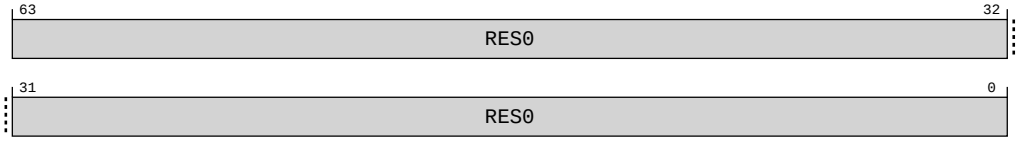


Table B-289: ERR0MISC1 bit descriptions

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:

RW

B.8.7 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

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-144: EXT_ERR0MISC2 bit assignments

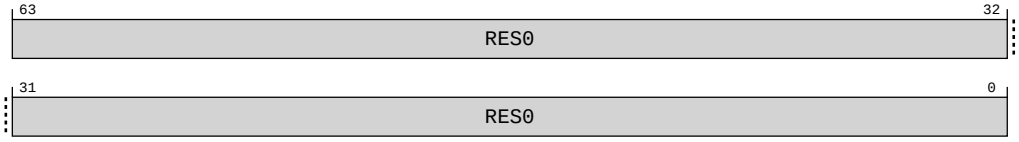


Table B-291: ERR0MISC2 bit descriptions

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.

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:

RW

B.8.8 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.

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-145: EXT_ERR0MISC3 bit assignments

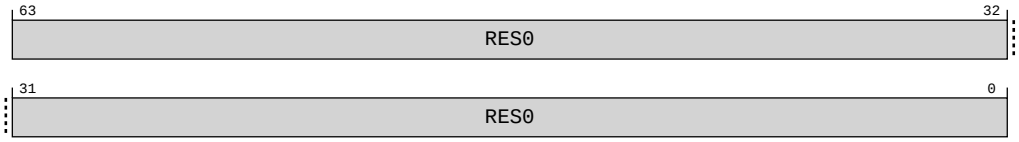


Table B-293: ERR0MISC3 bit descriptions

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

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:

RW

B.8.9 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

Component

RAS

Register offset

0x800

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0100 0000 0000 0000 0000 0000 0110 0010
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Bit descriptions

Figure B-146: EXT_ERR0PFGF bit assignments

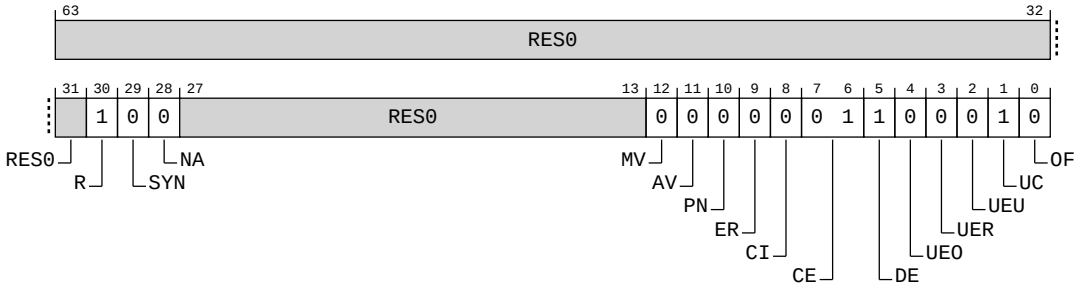


Table B-295: ERR0PFGF bit descriptions

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 ERR<n>PFGCTL.R.	0b1
[29]	SYN	Syndrome. Fault syndrome injection. **0b0** When an injected error is recorded, the node sets ERR0STATUS.{IERR, SERR} to **IMPLEMENTATION DEFINED** values. ERR0STATUS.{IERR, SERR} are **UNKNOWN** when ERR0STATUS.V is 0.	0b0
[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. **0b0** The component fakes detection of the error on an access to the component.	0b0
[27:13]	**RES0**	Reserved	**RES0**
[12]	MV	Miscellaneous syndrome. Defines whether software can control all or part of the syndrome recorded in the ERR<n>MISC<m> registers when an injected error is recorded. **0b0** When an injected error is recorded, the node might update the ERR<n>MISC<m> registers: • If any syndrome is recorded by the node in the ERR<n>MISC<m> registers, then ERR<n>STATUS.MV is set to 1. • Otherwise, ERR<n>STATUS.MV is unchanged.	0b0
[11]	AV	Address syndrome. Defines whether software can control the address recorded in ERR0ADDR when an injected error is recorded. **0b0** When an injected error is recorded, the node might record an address in ERR<n>ADDR. If an address is recorded in ERR<n>ADDR, then ERR<n>STATUS.AV is set to 1. Otherwise, ERR<n>ADDR and ERR<n>STATUS.AV are unchanged.	0b0
[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 ERR<n>STATUS.PN to 0.	0b0

Bits	Name	Description	Reset
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the ERR0STATUS.ER status flag. **0b0** When an injected error is recorded, the node sets ERR<n>STATUS.ER according to the architecture-defined rules for setting the ER field.	0b0
[8]	CI	Critical Error flag. Describes how the fault generation feature of the node sets the ERR0STATUS.CI status flag. **0b0** The node does not support this type of flag. This behavior replaces the architecture-defined rules for setting the CI bit.	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 ERR<n>STATUS.CE to 0b10.	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.	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.	0b0
[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.	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.	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.	0b1
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the ERR0STATUS.OF status flag. **0b0** When an injected error is recorded, the node sets ERR<n>STATUS.OF according to the architecture-defined rules for setting the OF field	0b0

Accessibility

Component	Offset	Instance	Range
RAS	0x800	ERR0PFGF	None

This interface is accessible as follows:

RO

B.8.10 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.

Attributes

Width

64

Component

RAS

Register offset

0x808

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 00xx xxxx xxxx xxxx xxx1 x0x0 000x xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-147: EXT_ERR0PFGCTL bit assignments

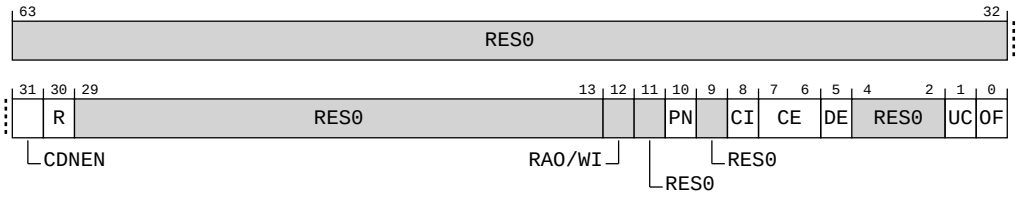


Table B-297: ERR0PFGCTL bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	CDNEN	Countdown Enable. Controls transfers from the value that is held in the ERXPFGCDN_EL1 into 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 `0b1` to this bit, the Error Generation Counter is set to ERXPFGCDN_EL1.CDN. Cold reset only. Unaffected by Warm reset	`0b0`
[30]	R	Restart. Controls whether, upon reaching zero, the Error Generation Counter restarts from the ERXPFGCDN_EL1 value or stops. **0b0** On reaching 0, the Error Generation Counter will stop. **0b1** On reaching 0, the Error Generation Counter is set to ERXPFGCDN_EL1.CDN. Cold reset only. Unaffected by Warm reset	`0b0`
[29:13]	**RES0**	Reserved	**RES0**
[12]	**RAO/WI**	Reserved	**RAO/WI**
[11]	**RES0**	Reserved	**RES0**
[10]	PN	Poison flag. The value that is written to ERXSTATUS.PN when an injected error is recorded. **0b00** ERXSTATUS.PN is set to 0 when an injected error is recorded. **0b01** ERXSTATUS.PN is set to 1 when an injected error is recorded.	`0b0`
[9]	**RES0**	Reserved	**RES0**
[8]	CI	Critical Error flag. The value that is written to ERXSTATUS.CI when an injected error is recorded. **0b0** ERXSTATUS.CI is set to 0 when an injected error is recorded. **0b1** ERXSTATUS.CI is set to 1 when an injected error is recorded.	`0b0`
[7:6]	CE	Corrected Error generation enable. Controls the type of Corrected Error condition that might be generated. The possible values are: **0b00** No error of this type will be generated. **0b01** A non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == `0b10`, might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	`0b00`

Bits	Name	Description	Reset
[5]	DE	Deferred Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0
[4:2]	**RES0**	Reserved	**RES0**
[1]	UC	Uncontainable Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0
[0]	OF	Uncontainable Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0

Accessibility

Component	Offset	Instance	Range
RAS	0x808	ERR0PFGCTL	None

This interface is accessible as follows:

RW

B.8.11 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

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-148: EXT_ERR0PFGCDN bit assignments

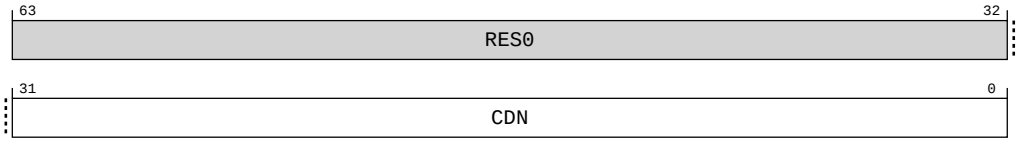


Table B-299: ERR0PFGCDN bit descriptions

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 ERXPFGCTL_EL1.CDNEN with 1. • The Error Generation Counter decrements to zero and ERXPFGCTL_EL1.R == 0b1. Unaffected by Cold or Warm reset. **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:

RW

B.8.12 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 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 000x
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
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-149: EXT_ERRGSR bit assignments

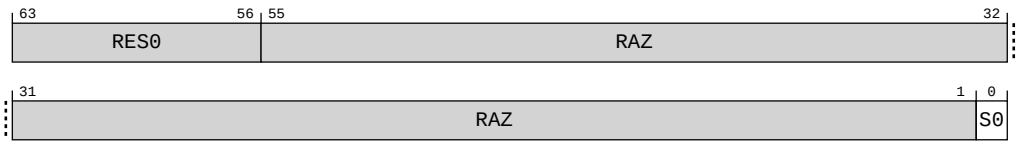


Table B-301: ERRGSR bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[55:1]	**RAZ**	Reserved	**RAZ**
[0]	S0	The status for error record 0. A read-only copy of ERR0STATUS.V. **0b0** No error. **0b1** One or more errors.	x

Accessibility

Component	Offset	Instance	Range
RAS	0xE00	ERRGSR	None

This interface is accessible as follows:

RO

B.8.13 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 1100 0001 0000 1000 0011 1011
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-150: EXT_ERRIIDR bit assignments

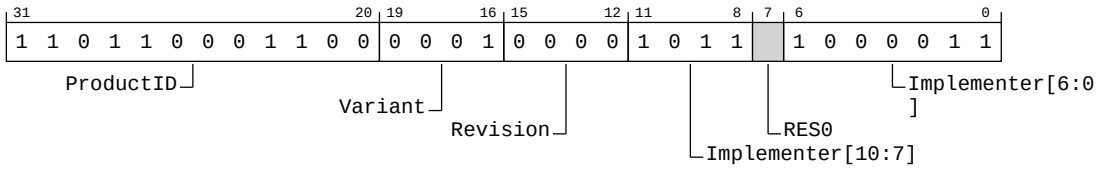


Table B-303: ERRIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	Part number, bits [11:0]. The part number is selected by the designer of the component. **0xD8C** C1-Ultra	0xD8C
[19:16]	Variant	Component major revision. This field distinguishes product variants or major revisions of the product. **0b0001** r1p0	0b0001
[15:12]	Revision	Component minor revision. This field distinguishes minor revisions of the product. **0b0000** r1p0	0b0000
[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. **0b10000111011** Arm Limited	0b10000111011
[7]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Range
RAS	0xE10	None

This interface is accessible as follows:

RO

B.8.14 ERRDEVAFF, Device Affinity Register

For a group of error records that has affinity with a single PE or a group of PEs, ERRDEVAFF is a copy of MPIDR_EL1 or part of MPIDR_EL1:

- If the group of error records has affinity with a single PE, the affinity level is 0, then ERRDEVAFF reads the same value as MPIDR_EL1, and ERRDEVAFF.F0V reads-as-one to indicate affinity level 0.
- If the group of error records has affinity with a group of PEs, the affinity level is 1, 2, or 3, then parts of ERRDEVAFF reads the same value as parts of MPIDR_EL1, and the rest of ERRDEVAFF indicates the level.

For example, if the group of PEs is a subset of the PEs at affinity level 1 then all of the following are true:

- All the PEs in the group have the same values in MPIDR_EL1.{Aff3,Aff2}, and these values are equal to ERRDEVAFF.{Aff3,Aff2}.
- ERRDEVAFF.Aff1 is nonzero and not `0x80`, and ERRDEVAFF.{Aff0,F0V} read-as-zero, to indicate at least affinity level 1. The subset of PEs at level 1 that the group of error records has affinity with is indicated by the least-significant set bit in ERRDEVAFF.Aff1. In this example, if ERRDEVAFF.Aff1[2:0] is `0b100`, then the group of error records has affinity with the up-to 8 PEs that have MPIDR_EL1.Aff1[7:3] == ERRDEVAFF.Aff1[7:3].

Depending on the **IMPLEMENTATION DEFINED** nature of the system, it might be possible that ERRDEVAFF is read before system firmware has configured the group of error records or the PE or group of PEs that the group of error records has affinity with. When this is the case, ERRDEVAFF reads as zero.

If RAS System Architecture v1.1 is not implemented then ERRDEVAFF can only describe a group of error records that is affine with a single PE or all the PEs at an affinity level.

Configurations

ERRDEVAFF is implemented only as part of a memory-mapped group of error records.

This register is present only when the group of error records has affinity with a PE or cluster of PEs. Otherwise, direct accesses to ERRDEVAFF are UNDEFINED.

Attributes

Width

64

Component

RAS

Register offset

0xFA8

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 xxxx xxxx 1x00 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-151: EXT_ERRDEVAFF bit assignments

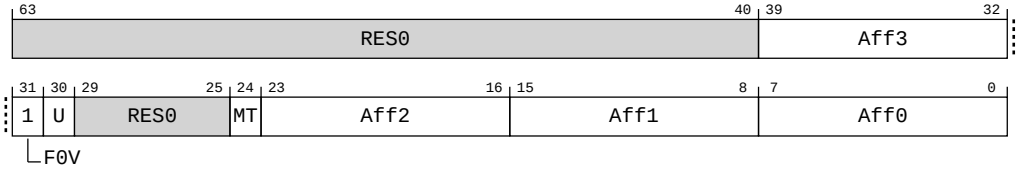


Table B-305: ERRDEVAFF bit descriptions

Bits	Name	Description	Reset
[63:40]	**RES0**	Reserved	**RES0**
[39:32]	Aff3	PE affinity level 3. The MPIDR_EL1.Aff3 field, viewed from the highest Exception level of the associated PE or PEs.	`8{x}`
[31]	F0V	Indicates that the ERRDEVAFF.Aff0 field is valid. **0b1** ERRDEVAFF.Aff0 is valid, and the PE affinity is at level 0.	`0b1`
[30]	U	Uniprocessor. The MPIDR_EL1.U field, viewed from the highest Exception level of the associated PE.	`x`
[29:25]	**RES0**	Reserved	**RES0**
[24]	MT	Multithreaded. The MPIDR_EL1.MT field, viewed from the highest Exception level of the associated PE.	`x`

Bits	Name	Description	Reset
[23:16]	Aff2	**When !IsZero(ext-ERRDEVAFF.[Aff1,Aff0,F0V])** PE affinity level 2. The MPIDR_EL1.Aff2 field, viewed from the highest Exception level of the associated PE or PEs. **Otherwise** PE affinity level 2. Defines part of the MPIDR_EL1.Aff2 field, viewed from the highest Exception level of the associated PEs. **0bxxxxxxx1** PE affinity is the subset of level 2 where ERRDEVAFF.Aff2[7:1] is the value of MPIDR_EL1.Aff2[7:1], viewed from the highest Exception level of the associated PEs. **0bxxxxxx10** PE affinity is the subset of level 2 where ERRDEVAFF.Aff2[7:2] is the value of MPIDR_EL1.Aff2[7:2], viewed from the highest Exception level of the associated PEs. **0bxxxxx100** PE affinity is the subset of level 2 where ERRDEVAFF.Aff2[7:3] is the value of MPIDR_EL1.Aff2[7:3], viewed from the highest Exception level of the associated PEs. **0bxxxx1000** PE affinity is the subset of level 2 where ERRDEVAFF.Aff2[7:4] is the value of MPIDR_EL1.Aff2[7:4], viewed from the highest Exception level of the associated PEs. **0bxxx10000** PE affinity is the subset of level 2 where ERRDEVAFF.Aff2[7:5] is the value of MPIDR_EL1.Aff2[7:5], viewed from the highest Exception level of the associated PEs. **0bxx100000** PE affinity is the subset of level 2 where ERRDEVAFF.Aff2[7:6] is the value of MPIDR_EL1.Aff2[7:6], viewed from the highest Exception level of the associated PEs. **0bx1000000** PE affinity is the subset of level 2 where ERRDEVAFF.Aff2[7] is the value of MPIDR_EL1.Aff2[7], viewed from the highest Exception level of the associated PEs. **0x80** PE affinity is at level 3.	8{x}

Bits	Name	Description	Reset
[15:8]	Aff1	**When !IsZero(ext-ERRDEVAFF.[Aff0,F0V])** PE affinity level 1. The MPIDR_EL1.Aff1 field, viewed from the highest Exception level of the associated PE or PEs. **Otherwise** PE affinity level 1. Defines part of the MPIDR_EL1.Aff1 field, viewed from the highest Exception level of the associated PEs. **0bxxxxxxx1** PE affinity is the subset of level 1 where ERRDEVAFF.Aff1[7:1] is the value of MPIDR_EL1.Aff1[7:1], viewed from the highest Exception level of the associated PEs. **0bxxxxxx10** PE affinity is the subset of level 1 where ERRDEVAFF.Aff1[7:2] is the value of MPIDR_EL1.Aff1[7:2], viewed from the highest Exception level of the associated PEs. **0bxxxxx100** PE affinity is the subset of level 1 where ERRDEVAFF.Aff1[7:3] is the value of MPIDR_EL1.Aff1[7:3], viewed from the highest Exception level of the associated PEs. **0bxxxx1000** PE affinity is the subset of level 1 where ERRDEVAFF.Aff1[7:4] is the value of MPIDR_EL1.Aff1[7:4], viewed from the highest Exception level of the associated PEs. **0bxxx10000** PE affinity is the subset of level 1 where ERRDEVAFF.Aff1[7:5] is the value of MPIDR_EL1.Aff1[7:5], viewed from the highest Exception level of the associated PEs. **0bxx100000** PE affinity is the subset of level 1 where ERRDEVAFF.Aff1[7:6] is the value of MPIDR_EL1.Aff1[7:6], viewed from the highest Exception level of the associated PEs. **0bx1000000** PE affinity is the subset of level 1 where ERRDEVAFF.Aff1[7] is the value of MPIDR_EL1.Aff1[7], viewed from the highest Exception level of the associated PEs. **0x80** PE affinity is at level 2. **0x00** PE affinity is above level 2 or a subset of level 2.	8{x}
[7:0]	Aff0	PE affinity level 0. The MPIDR_EL1.Aff0 field, viewed from the highest Exception level of the associated PE.	8{x}

Accessibility

Component	Offset	Instance	Range
RAS	0xFA8	ERRDEVAFF	None

This interface is accessible as follows:

RO

B.8.15 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 xxxx 0000 1010 0000 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-152: EXT_ERRDEVARCH bit assignments

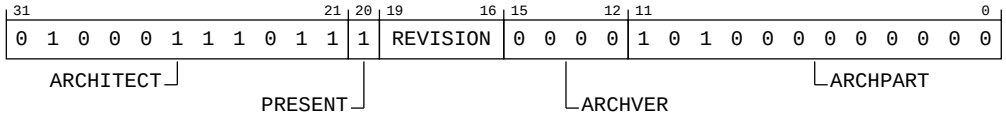


Table B-307: ERRDEVARCH bit descriptions

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. **`0b0000`** RAS System Architecture, error record group v1.0. **`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.	The reset values can be the following: `0b0000`, `0b0001`, respective to the value.
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **`0b0000`** RAS System Architecture, error record group v1.	`0b0000`
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **`0xA00`** RAS System Architecture, error record group.	`0xA00`

Accessibility

Component	Offset	Instance	Range
RAS	0xFBC	ERRDEVARCH	None

This interface is accessible as follows:

RO

B.8.16 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-153: EXT_ERRDEVID bit assignments

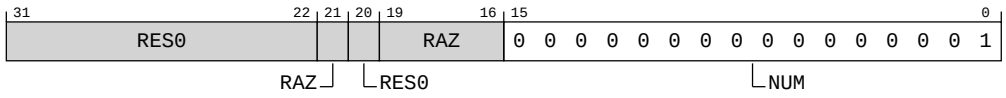


Table B-309: ERRDEVID bit descriptions

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.8.17 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.

Attributes

Width

32

Component

RAS

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-154: EXT_ERRPIDR4 bit assignments

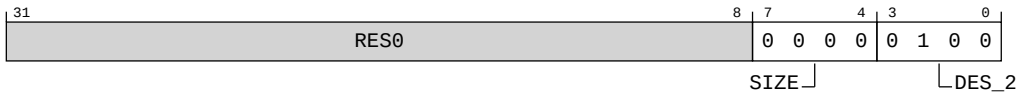


Table B-311: ERRPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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 $2^{ERRPIDR4.SIZE}$ 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:

RO

B.8.18 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.

Attributes

Width

32

Component

RAS

Register offset

0xFE0

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1000 1100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-155: EXT_ERRPIDR0 bit assignments

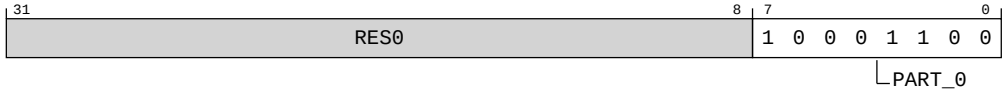


Table B-313: ERRPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[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. **0x8C** C1-Ultra	0x8C

Accessibility

Component	Offset	Instance	Range
RAS	0xFE0	ERRPIDR0	None

This interface is accessible as follows:

RO

B.8.19 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.

Attributes

Width

32

Component

RAS

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-156: EXT_ERRPIDR1 bit assignments

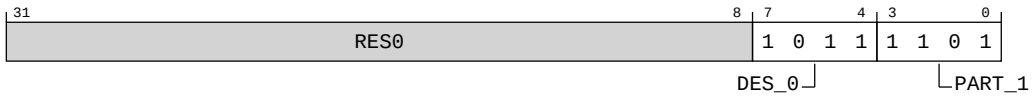


Table B-315: ERRPIDR1 bit descriptions

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

Bits	Name	Description	Reset
[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-Ultra	0b1101

Accessibility

Component	Offset	Instance	Range
RAS	0xFE4	ERRPIDR1	None

This interface is accessible as follows:

RO

B.8.20 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.

Attributes

Width

32

Component

RAS

Register offset

0xFE8

Access type

RO

Reset value

When the component uses a 12-bit part number

0000 0000 0000 0000 0000 0000 0001 1011

When the component uses a 16-bit part number

0000 0000 0000 0000 0000 0000 xxxx 1011

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When the component uses a 12-bit part number

Figure B-157: EXT_ERRPIDR2 bit assignments

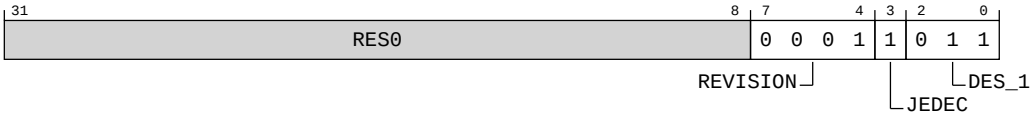


Table B-317: ERRPIDR2 bit descriptions

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`** r1p0	`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`

When the component uses a 16-bit part number

Figure B-158: EXT_ERRPIDR2 bit assignments

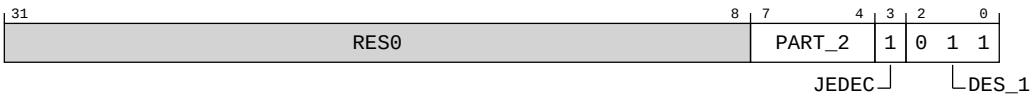


Table B-318: ERRPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	PART_2	Part number, bits [15:12]. 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.	xxxx
[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:

RO

B.8.21 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.

Attributes

Width

32

Component

RAS

Register offset

0xFEC

Access type

RO

Reset value

When the component uses a 12-bit part number

0000 0000 0000 0000 0000 0000 0000 0000

When the component uses a 16-bit part number

0000 0000 0000 0000 0000 0000 xxxx 0000


Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When the component uses a 12-bit part number

Figure B-159: EXT_ERRPIDR3 bit assignments

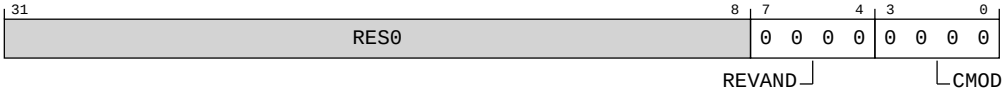


Table B-320: ERRPIDR3 bit descriptions

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. **`0b0000`** r1p0	`0b0000`
[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`

When the component uses a 16-bit part number

Figure B-160: EXT_ERRPIDR3 bit assignments

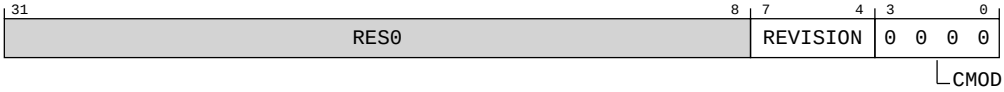


Table B-321: ERRPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Component revision. When a component is changed, ERRPIDR3.REVISION is increased to ensure that software can differentiate the different revisions of the component.	xxxx
[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

This interface is accessible as follows:

RO

B.8.22 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.

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-161: EXT_ERRCIDR0 bit assignments

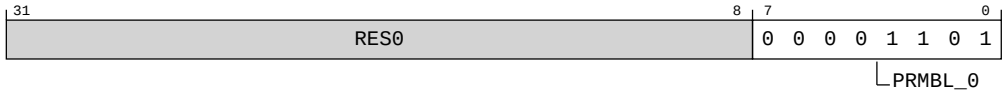


Table B-323: ERRCIDR0 bit descriptions

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:

RO

B.8.23 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.

Attributes

Width

32

Component

RAS

Register offset

0xFF4

Access type

RO

Reset value

```
0000 0000 0000 0000 0000 0000 1111 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Bit descriptions

Figure B-162: EXT_ERRCIDR1 bit assignments

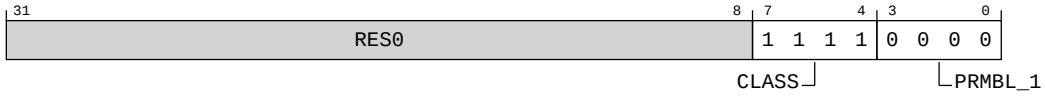


Table B-325: ERRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1111** Generic peripheral with **IMPLEMENTATION DEFINED** register layout.	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:

RO

B.8.24 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*.

Configurations

ERRCIDR2 is implemented only as part of a memory-mapped group of error records.

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-163: EXT_ERRCIDR2 bit assignments

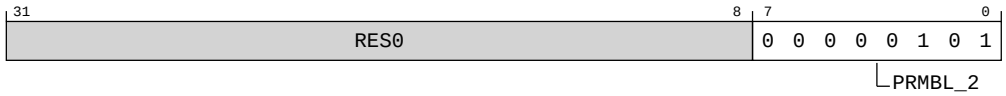


Table B-327: ERRCIDR2 bit descriptions

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:

RO

B.8.25 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.

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-164: EXT_ERRCIDR3 bit assignments

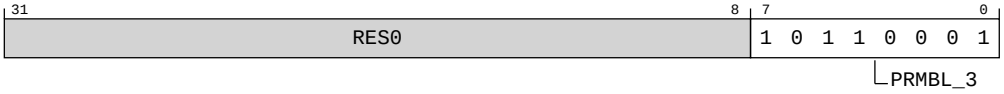


Table B-329: ERRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[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:

RO

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.

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 r1p0, 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
0100-04	10 September 2025	Non-Confidential	Second early access release for r1p0
0100-03	30 August 2024	Confidential	First early access release for r1p0
0000-02	29 February 2024	Confidential	First limited access release for r0p0
0000-01	15 December 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 897.

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 convention	Throughout document
Updated descriptions and reorganized supported standards and features tables.	1.4 Supported standards, specifications, and features on page 21
Updated details of core components figure	2.1 Core components on page 36
Updated the Warm Reset power state description in the core power modes table.	4.4 Core power modes on page 47
Added 'L1 Statistical Profiling Extension TLB' component information to the MMU components table and additional L1 TLB Functionality information following the table.	5.1 Memory Management Unit components on page 56
Updated and expanded entries in the RAS registers summary table.	10.7 External RAS registers on page 98
Removed IMP_CME* registers from the Generic System Control registers summary table as they may be found in CME documentation.	System control
Removed CTI register identification values topic from the Debug chapter as this is now documented by the DSU.	Debug
Removed Effects of resets on debug registers topic that referenced COMPLEXRESET_N/ COMPLEXPOR_RESET_N as they have been removed from the top-level interface.	Debug register interfaces
Assorted updates to PMU events including the addition of new events: SVE_SPEC, ASE_SVE_SPEC, and SVE_UNPRED_SPEC, SVE_LDST_SPEC, SVE_LD_SPEC, SVE_ST_SPEC, and SVE_PRF_SPEC. Also added SE_SPEC, SME_INST_SPEC, SE_INST_SPEC, and ZA_ACTIVE_CYCLES.	17.1 Common PMU events on page 125
Updated descriptions and values for IMP_L2_CHI*. Removed IMP_STALL_BACKEND_CME and IMP_OP_CME_ISSUE. Added 23 PMU events from 0x3200 through 0x3238 describing STALL, CME, and SSVE events.	17.2 Implementation-defined PMU events on page 158
Added AMU events for CPU_ACTIVITY, STALL_BACKEND_BUSY_CME, and STALL_BACKEND_BUSY_CME_ARB.	20.3 Activity monitors events on page 193
Added 3 AMEVCNTR* and 3 AMEVTYPER* Activity Monitor registers.	20. Activity Monitors Extension support on page 192
Updates to register information throughout.	A. AArch64 registers on page 202 and B. External registers on page 622

Table 4: Differences between Issue 0000-02 and 0100-03

Change	Location
First early access release for r1p0	—
Editorial changes	Throughout document
Restructured and renamed the "Supported standards, specifications, and features" section.	1.4 Supported standards, specifications, and features on page 21
Caution and Note added; clarified use of WARM_RST mode; added WFI information.	4.4.6 Warm reset mode on page 51
Updated VA values.	5.3 Translation Lookaside Buffer match process on page 58
Updated information on contiguous hints, conflict aborts, and conflict abort exceptions.	5.6 Responses on page 60
Clarified details of n prefix note.	5.7 Memory behavior and supported memory types on page 62
Corrected register names in the "System registers used to access internal memory" table. Added information regarding DSB and ISB instructions.	9. Direct access to internal memory on page 76
Clarified the cache protection capabilities of core RAMs	10.1 Cache protection behavior on page 95
Updated the CoreSight Peripheral Identification (ID) in the "CoreSight component ID" table. Updated the Debug DevArch value in the *CoreSight component ID* table.	16.6 CoreSight component identification on page 119
Assorted updates to the following PMU events: L1D_CACHE_REFILL, LDST_SPEC, L2D_CACHE_PRFM, L2I_CACHE_PRFM, L2I_CACHE_REFILL_PRFM, L2I_CACHE_HIT_RD, L2D_CACHE_HIT_RW_FHWPRF, L2I_CACHE_HIT_PRFM, IMP_L2D_TLB_REFILL_PRFM, IMP_STALL_BACKEND_RENAME_PDRF (new event number), IMP_STALL_BACKEND_CPUBOUND_OTHER (new event number), IMP_STALL_FRONTEND_SPEC_THROT (new), IMP_STALL_BACKEND_SPEC_THROT (new), STALL_BACKEND_MEM_CME.	17. Performance Monitors Extension support on page 125
Added PMIIDR register information.	17.6 External PMU registers summary on page 174
Removed note regarding Access enable bit.	20.1 Activity monitors access on page 192
Updated the "SPE events packet" table with new bit values [25,24,22] and associated definitions.	21.1 Statistical Profiling Extension events packet on page 199
Added CMC_MIN_TRAIN_FREQ to IMP_CPUECTLR2_EL1 bit descriptions table.	A.4.11 IMP_CPUECTLR2_EL1, CPU Extended Control Register 2 on page 320
Updated the accessibility code block for IMP_CPUL2DIRTYLNCT_EL1 register.	A.4.12 IMP_CPUL2DIRTYLNCT_EL1, CPU L2 Dirty Line Count Register on page 325
Updated WFE_RET_CTRL and WFI_RET_CTRL values in the IMP_CPUPWRCTLR_EL1 register bit description.	A.4.14 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 328

Change	Location
Clarified the L2DIRTYEN bit write-accessibility content in the ACTLR_EL3 register.	A.4.34 ACTLR_EL3, Auxiliary Control Register (EL3) on page 374
Added ACTM_EN (Activity Meter Enable) to the IMP_CPUPPMCR_EL3 register.	A.9.2 IMP_CPUPPMCR_EL3, Global Performance and Power Management Configuration Register on page 506
Specific reset values added to various registers in register summary tables and register descriptions.	Throughout document.

Table 5: Differences between Issue 0100-03 and 0100-04

Change	Location
Second early access release for r1p0	—
Editorial changes	Throughout document
Updated note on identical configurations	1.2 C1-Ultra core configuration options on page 19
Updated and reordered standards and specifications content	1.4 Supported standards, specifications, and features on page 21
Clarifications to Off power mode	4.4.2 Off mode on page 49
Added additional detail on Maximum Power Mitigation Mechanism	4.5.1 Maximum Power Mitigation Mechanism on page 52
Updates and clarifications to the core powerup and powerdown sequence	4.6 C1-Ultra core powerup and powerdown sequence on page 53
Note added regarding cache maintenance operations	6.1 L1 instruction cache behavior on page 65
Updates to L1 data cache tag location encoding and L1 data cache data location encoding	9.1 L1 cache encodings on page 77
Adjusted values in L2 tag cache format for Data Register 0	9.2.1 L2 tag RAM returned data on page 87
Updated PMU event names and definitions	17.1 Common PMU events on page 125 17.2 Implementation-defined PMU events on page 158
Updated AMU auxiliary counter numbers from three to six (10-15)	20.2 Activity monitors counters on page 193
Updated information for various registers	A.1.11 AMEVTYPER14_EL0, Activity Monitors Event Type Registers 1 on page 223 A.1.12 AMEVTYPER15_EL0, Activity Monitors Event Type Registers 1 on page 225 A.4.1 ACTLR_EL1, Auxiliary Control Register (EL1) on page 290 A.5.5 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1 on page 409 A.5.22 CLIDR_EL1, Cache Level ID Register on page 449 A.11.2 PMSEVFR_EL1, Sampling Event Filter Register on page 576

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 underline`	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**.

We recommend the following. If you do not follow these recommendations your system might not work.

Your system requires the following. If you do not follow these requirements your system will not work.

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.

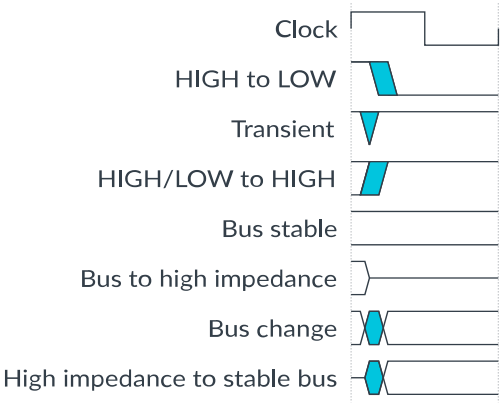
Remember — This information reminds you of something important relating to the current content.

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.

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`.

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-DynamIQ™ Shared Unit Configuration and Integration Manual	107805	Confidential
Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual	107804	Non-Confidential
Arm® C1-Ultra and C1-Premium (MP201) Release Note	109954	Confidential
Arm® C1-Ultra Core Configuration and Integration Manual	108015	Confidential
Arm® C1-Ultra and C1-Premium Cryptographic Extension (MP202) Release Note	109955	Confidential
Arm® C1-Ultra Core Cryptographic Extension Technical Reference Manual	108027	Non-Confidential
Arm® C1-Ultra Core Telemetry Specification	109820	Non-Confidential
Arm® C1-Scalable Matrix Extension 2 Configuration and Integration Manual	107832	Confidential
Arm® C1-Scalable Matrix Extension 2 Technical Reference Manual	107831	Non-Confidential
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual	101088	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® CoreSight™ Architecture Specification v3.0	IHI 0029	Non-Confidential
Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4	IHI 0069	Non-Confidential
Arm® Reliability, Availability, and Serviceability (RAS) System Architecture	IHI 0100	Non-Confidential