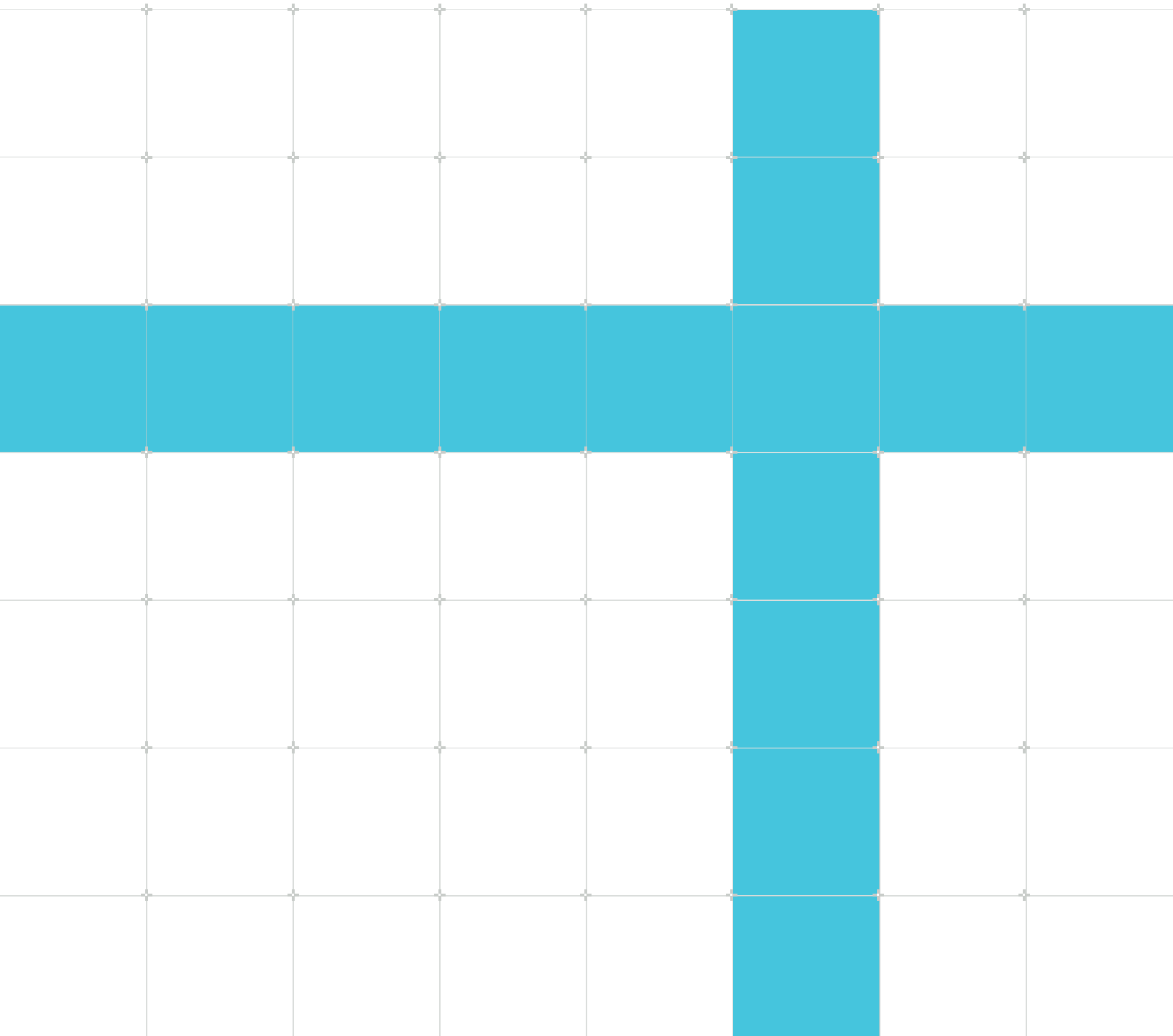
Arm® Cortex-A520 Core

Revision: r0p2

Technical Reference Manual

Non-Confidential

Issue 05
102517_0002_05_en



Arm® Cortex-A520 Core

Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-01	15 November 2021	Confidential	First beta release for r0p0
0000-02	8 April 2022	Confidential	First limited access release for r0p0
0001-03	29 July 2022	Confidential	First early access release for r0p1
0001-04	29 May 2023	Non-Confidential	Second early access release for r0p1
0002-05	15 December 2023	Non-Confidential	First early access release for r0p2

Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Feedback

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

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

Inclusive language commitment

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

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

To report offensive language in this document, email terms@arm.com.

Contents

1. Introduction

1.1 Product revision status

The r*x*p*y* identifier indicates the revision status of the product described in this manual, for example, r1p2, where:

r*x* Identifies the major revision of the product, for example, r1.

p*y* Identifies the minor revision or modification status of the product, for example, p2.

1.2 Intended audience

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

1.3 Conventions

The following subsections describe conventions used in Arm documents.

Glossary

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

See the Arm Glossary for more information: developer.arm.com/glossary.

Convention	Use
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>`

Convention	Use
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.

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

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

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

An important piece of information that needs your attention.

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

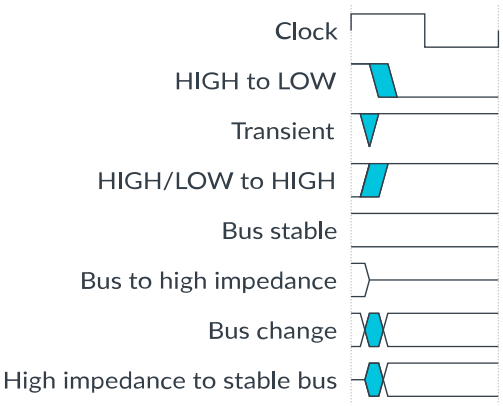
A reminder of something important that relates to the information you are reading.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

Figure 1-1: Key to timing diagram conventions



Signals

The signal conventions are:

Signal level

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n

At the start or end of a signal name, n denotes an active-LOW signal.

Register descriptions

Reset definitions

Replication Operator {}

Verilog replication operators are used for reset values over 8-bits.

For example, `{16{1'b0}}` indicated a binary value of 16 zeros.

x

Resets that are unknown are indicated with `x`.

1.4 Useful resources

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

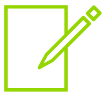
Access to Arm documents depends on their confidentiality:

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

- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual	102547	Non-Confidential
Arm® Cortex-A520 Core Configuration and Integration Manual	102518	Confidential
Arm® Cortex-A520 Core Cryptographic Extension Technical Reference Manual	102519	Non-Confidential
Arm®DynamIQ™ Shared Unit-120 Configuration and Integration Manual	102548	Confidential
Arm® Cortex-A520 Core Release Note	-	Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® 5 CHI Architecture Specification	IHI 0050	Non-Confidential
Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture	DDI 0598	Non-Confidential
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487	Non-Confidential
Arm® CoreSight™ Architecture Specification v3.0	IHI 0029	Non-Confidential
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Configuration and Integration Manual	101089	Confidential
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual	101088	Non-Confidential
Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4	IHI 0069	Non-Confidential
Arm® Architecture Reference Manual Supplement, Reliability, Availability, and Serviceability (RAS), for A-profile architecture	DDI 0587	Non-Confidential


Note

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

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

2. The Cortex-A520 core

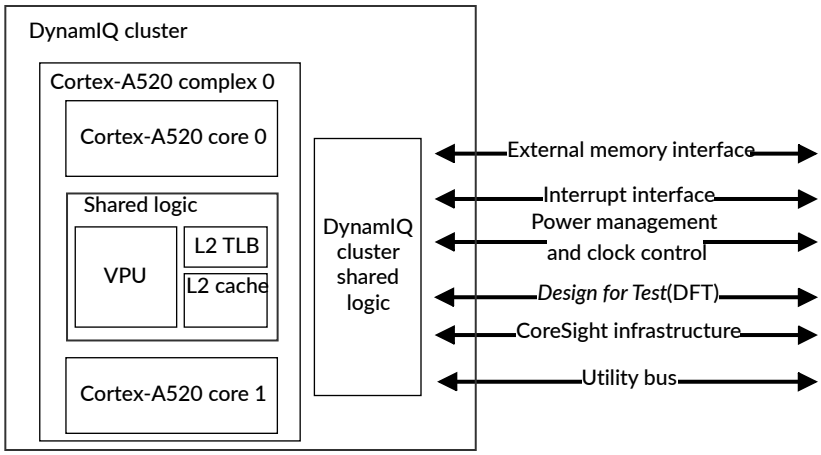
The Cortex-A520 core is a high-efficiency and low-power core that implements the Arm®v9.2-A architecture. The Arm®v9.2-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.7-A.

The Cortex-A520 core is implemented inside a DSU-120 DynamIQ™ cluster. It is connected to the *DynamIQ Shared Unit-120* 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 Cortex-A520 core is the high-efficiency core.

Cortex-A520 cores are implemented inside a block called a complex, which contains up to two cores. Within a dual-core complex, the *Vector Processing Unit* (VPU), the L2 *Translation Lookaside Buffer* (TLB), and the L2 cache logic are shared between cores.

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

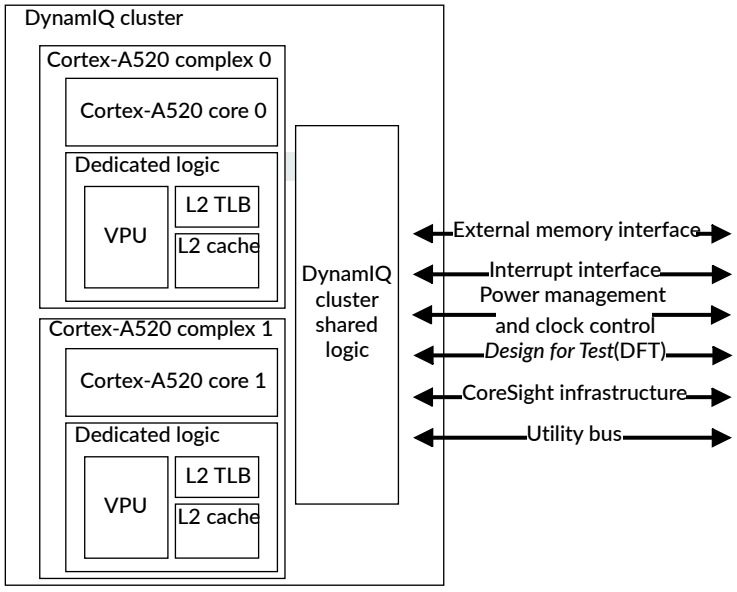
Figure 2-1: Example configuration with a Cortex-A520 dual-core complex



You can also configure a complex that contains a single Cortex-A520 core with dedicated logic. You can configure your systems so that all cores are configured in single-core complexes. This type of configuration improves performance but at the cost of area efficiency.

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

Figure 2-2: Example configuration with two Cortex-A520 single-core complexes



Note

- This manual applies to the Cortex-A520 core only. Read this manual together with the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for detailed information about the DSU-120.
- This manual does not provide a complete list of registers. Read this manual together with the *Arm® Architecture Reference Manual for A-profile architecture*.

2.1 Cortex-A520 core features

You can use the Cortex-A520 core in a standalone DynamIQ™ configuration where your homogenous DSU-120 DynamIQ™ cluster includes one or more Cortex-A520 complexes. You can also use the Cortex-A520 core as the high-efficiency core in a heterogenous cluster.

Regardless of the cluster configuration, the Cortex-A520 core always has the same features as described in the following lists.

Core features

- Implementation of the Arm®v9.2-A A64 instruction set
- AArch64 Execution state at all Exception levels, EL0 to EL3
- Separate L1 data and instruction side memory systems with a *Memory Management Unit* (MMU)
- 40-bit *Physical Address* (PA) and 48-bit *Virtual Address* (VA)
- In-order pipeline with direct and indirect branch prediction

- *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
- *Scalable Vector Extension* (SVE) and SVE2 *Single Instruction Multiple Data* (SIMD) instruction set, offering Advanced SIMD and floating-point architecture support
- *Activity Monitoring Unit* (AMU)
- Support for the optional Cryptographic Extension

The Cryptographic Extension is licensed separately.

Cache features

- Separate L1 data and instruction caches
- Optional unified L2 cache
- L1 and L2 cache protection with *Error Correcting Code* (ECC) or parity
- Optional error protection with parity or *Error Correcting Code* (ECC) allowing:
  - *Single Error Correction and Double Error Detection* (SECDED) on L1 data cache and L2 cache, and L2 *Translation Lookaside Buffer* (TLB)
  - *Single Error Detection* (SED) on L1 instruction cache
- Support for *Memory System Resource Partitioning and Monitoring* (MPAM)

Debug features

- Arm®v9.2-A debug logic
- *Performance Monitoring Unit* (PMU)
- *Embedded Trace Extension* (ETE)
- *TRace Buffer Extension* (TRBE)
- Optional *Embedded Logic Analyzer* (ELA), ELA-600

The ELA-600 is licensed separately.

Related information

3. Technical overview on page 37

2.2 Cortex-A520 core configuration options

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

Some of these options:

- Apply to an individual Cortex-A520 complex
- Must be the same across all Cortex-A520 cores in the DSU-120 DynamIQ™ cluster
- Must be the same across all cores of all types in the DSU-120 DynamIQ™ cluster

Cortex-A520 core configuration options include:

Dual or single core

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

Cryptographic Extension

You can configure your implementation with or without the Cryptographic Extension. There must be consistency in setting this parameter to the same value for all cores. The Cryptographic Extension is an optional separately licensable product.

Vector datapath size

The size of the vector datapaths can be 2×64-bit or 2×128-bit. This is a per-complex option.

Number of PMU event counters

The number of event counters in the *Performance Monitoring Unit* (PMU) can be six or 20. The selected option applies to all Cortex-A520 cores in the DSU-120 DynamIQ™ cluster.

Core cache protection

Cache protection using ECC/parity. Configure whether your core implementation includes cache protection. The selected option applies to all cores in the DynamIQ™ cluster, including non-Cortex-A520 cores.

CoreSight™ Embedded Logic Analyzer

Optionally, you can include support for integrating CoreSight™ ELA-600, as a separately licensable product. This is a per-complex option.

ELA ATB FIFO depth

If the *Embedded Logic Analyzer* (ELA) is included, the depth of the ATB FIFO in the ELA can be 4, 8, 16, 32 or 64 entries. This is a per-complex option.

L1 instruction cache size

The L1 instruction cache can be 32KB or 64KB. The selected option applies to all Cortex-A520 cores in the DynamIQ™ cluster.

L1 data cache size

The L1 data cache can be 32KB or 64KB. The selected option applies to all Cortex-A520 cores in the DynamIQ™ cluster.

L2 cache

Configure whether the L2 cache is present. This is a per-complex option.

L2 cache size

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

L2 slices

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

L2 cache data RAM partitions

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

L2 doubled clock pulse

L2 clock pulse width can be optionally doubled. This supports meeting minimum clock pulse width requirements on some RAMs.

Evict/Allocate feature

Configure whether the *Evict/Allocate* (EVA) feature is used on the L2 cache data RAMs for all Cortex-A520 complexes.

See *RTL configuration process* in the *Arm® Cortex-A520 Core Configuration and Integration Manual* for detailed configuration options and guidelines.

2.3 DSU-120 dependent features

Some *DynamIQ Shared Unit-120* features and behaviors depend on whether your licensed core supports a particular feature.

The following table describes which DSU-120 dependent features are supported in your Cortex-A520 core.

Table 2-1: Cortex-A520 core features that have a dependency on the DSU-120

Feature	Supported in the Cortex-A520 core	Dependency on the DSU-120
Direct connect	No	-
Core included in a complex	Yes	Affects the DSU-120 DynamIQ™ cluster configuration and external signals. For more information on configurations, see 2.2 Cortex-A520 core configuration options on page 25.
Cryptographic Extension	Yes, as an option	Affects the external signals of the DSU-120. For more information on MPMM, see 5.6 Performance and power management on page 58.
Maximum Power Mitigation Mechanism (MPMM)	Yes	
Performance Defined Power (PDP) feature	No	
DISPBLKy Dispatch block signal	No	

Feature	Supported in the Cortex-A520 core	Dependency on the DSU-120
Statistical Profiling Extension (SPE) architecture	No	
Physical Address (PA) width	40-bit	Affects the CHI master and AXI master port bus widths. For more details, see the following chapters of the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*: • *CHI master interface* • *AXI master interface*


Note

- The Cryptographic Extension is supplied under a separate license.

2.4 Supported standards and specifications

The Cortex-A520 core complies with the Arm®v9.2-A architecture. The Arm®v9.2-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.7-A. The Cortex-A520 core also complies with specific Arm®v8-A architecture extensions and supports interconnect, interrupt, timer, debug, and trace architectures.

The Cortex-A520 core supports AArch64 only at Exception levels EL0 to EL3.

Not all architectural features are implemented in the Cortex-A520 core. The following tables show the implementation status of Arm®v8-A and Arm®v9-A features supported by the Cortex-A520 core. There is a separate table for each version of the Arm®v8-A and Arm®v9-A architectures.


Note

- Not all Arm®v8-A and Arm®v9-A architectural features are listed in the following tables. For more information on all the architectural features, see the *Arm® Architecture Reference Manual for A-profile architecture*.
- The Cortex-A520 core is compatible with the architecture for the *DynamIQ Shared Unit-120*. See the *Supported standards and specifications* section in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for a list of specific architectural versions and features supported by the DSU-120.

Table 2-2: Implementation status of the Arm®v8.0-A features in the Cortex-A520 core

Feature	Implemented	Description
FEAT_PCSRv8	No	PC Sample-based Profiling Extension

Feature	Implemented	Description
Cryptographic Extension	Yes Configurable	For more information and additional cryptographic register descriptions, see the *Arm® Cortex-A520 Core Cryptographic Extension Technical Reference Manual*. This extension is licensed separately and access to the documentation is restricted by contract with Arm.
FEAT_SHA1	Yes Configurable Supported as part of the Arm®v8-A Cryptographic Extension	Advanced SIMD SHA1 instructions
FEAT_SHA256		Advanced SIMD SHA256 instructions
FEAT_AES		Advanced SIMD AES instructions
FEAT_PMULL		Advanced SIMD PMULL instructions
FEAT_DoubleLock	No	Double Lock
FEAT_CP15SDISABLE2	No	CP15DISABLE2
FEAT_FP	Yes	Floating point extension
FEAT_AdvSIMD	Yes	Advanced SIMD Extension For more information and register descriptions, see 14. Advanced SIMD and floating-point support on page 98.
FEAT_CRC32	Yes	CRC32 instructions
FEAT_PMUv3	Yes	PMU extension version 3
FEAT_nTLBPA	Yes	No intermediate caching by output address in TLB
FEAT_SB	Yes	Speculation barrier
FEAT_SSBS	Yes	Speculative Store Bypass Safe Instruction
FEAT_CSV2	Yes	Cache Speculation Variant 2
FEAT_CSV2_1p1	No	Cache Speculation Variant 2 version 1.1
FEAT_CSV2_1p2	No	Cache Speculation Variant 2 version 1.2
FEAT_CSV2_2	Yes	Cache Speculation Variant 2 version 2
FEAT_CSV3	Yes	Cache Speculation Variant 3
FEAT_SPECRES	Yes	Speculation restriction instructions
FEAT_DGH	Yes	Data Gathering Hint
FEAT_ETS	Yes	Enhanced Translation Synchronization
FEAT_ECBHB	Yes	*Exploitative Control using Branch History Buffer* information between exception levels The branch history information created in a context before an exception to a higher exception level, using AArch64, cannot be used by code before that exception. This prevents exploitative control of the execution of any indirect branches in code in a different context after the exception.

Table 2-3: Implementation status of the Arm®v8.1-A features in the Cortex-A520 core

Feature	Implemented	Description
FEAT_LSE	Yes	Large System Extensions
FEAT_RDM	Yes	Rounding double multiply accumulate
FEAT_HPDS	Yes	Hierarchical permission disables in translation tables
FEAT_VHE	Yes	Virtualization Host Extensions

Feature	Implemented	Description
FEAT_PAN	Yes	Privileged access-never
FEAT_LOR	Yes	Limited ordering regions
FEAT_HAFDBS	Yes	Hardware updates to access flag and dirty state in translation tables
FEAT_VMID16	Yes	16-bit VMID
FEAT_PMUv3p1	Yes	PMU extensions version 3.1
FEAT_Debugv8p1	Yes	Debug with VHE
FEAT_PAN3	Yes	Support for SCTLR_ELx.EPAN

Table 2-4: Implementation status of the Arm®v8.2-A features in the Cortex-A520 core

Feature	Implemented	Description
FEAT_TTCNP	Yes	Common not private translations
FEAT_XNX	Yes	Execute-never control distinction by Exception level at stage 2
FEAT_UAO	Yes	Unprivileged Access Override control
FEAT_PAN2	Yes	AT S1E1R and AT S1E1W instruction variants for PAN
FEAT_DPB	Yes	DC CVAP instruction
FEAT_Debugv8p2	Yes	Arm®v8.2-A Debug
FEAT_IESB	Yes	Implicit Error synchronization event
FEAT_AA32HPD	No	AArch32 Hierarchical permission disables
FEAT_HPDS2	Yes	Hierarchical permission disables in translation tables 2
FEAT_LSMAOC	No	Load/Store instruction multiple atomicity and ordering controls
FEAT_FP16	Yes	Half-precision floating-point data processing
FEAT_LVA	No	Large VA support
FEAT_LPA	No	Large PA and IPA support
FEAT_VPIPT	No	VMID-aware PIPT instruction cache
FEAT_PCSRv8p2	Yes	PC Sample-based profiling version 8.2
FEAT_RAS	Yes	*Reliability, Availability, and Serviceability* (RAS) Extension version 1.1
FEAT_SPE	No	*Statistical Profiling Extension* (SPE)
FEAT_SVE	Yes	*Scalable Vector Extension* (SVE)
FEAT_SHA512	Yes	Advanced SIMD SHA512 instructions
FEAT_SHA3	Configurable	Advanced SIMD EOR3, RAX1, XAR, and BCAX instructions
FEAT_SM3	Configurable	Advanced SIMD SM3 instructions
FEAT_SM4	Supported as part of the Arm®v8-A Cryptographic Extension	Advanced SIMD SM4 instructions
FEAT_DotProd	Yes	Advanced SIMD Int8 dot product instructions
FEAT_FHM	Yes	Half-precision floating-point FMLAL instructions
FEAT_EVT	Yes	Enhanced Virtualization Traps
FEAT_DPB2	Yes	DC CVADP instruction
FEAT_BF16	Yes	AArch64 BFloat16 instructions
FEAT_AA32BF16	No	AArch32 BFloat16 instructions

Feature	Implemented	Description
FEAT_I8MM	Yes	Int8 Matrix Multiplication
FEAT_AA32I8MM	No	AArch32 Int8 Matrix Multiplication
FEAT_F32MM	No	SVE single-precision floating-point matrix multiply instruction
FEAT_F64MM	No	SVE double-precision floating-point matrix multiply instruction

Table 2-5: Implementation status of the Arm®v8.3-A features in the Cortex-A520 core

Feature	Implemented	Description
FEAT_PAuth	Yes	Pointer authentication
FEAT_EPAC	No	Enhanced Pointer authentication
FEAT_PACIMP	No	Pointer authentication - IMPLEMENTATION DEFINED algorithm
FEAT_PACQARMA5	No	Pointer authentication - QARMA5 algorithm
FEAT_PACQARMA3	Yes	Pointer authentication - QARMA3 algorithm
FEAT_CONSTPACFIELD	Yes	PAC Algorithm enhancement
FEAT_JSCVT	Yes	JavaScript FJCVTS conversion instruction
FEAT_NV	No	Nested virtualization
FEAT_LRCPC	Yes	Load-acquire RCpc instructions
FEAT_FCMA	Yes	Floating-point FCMLA and FCADD instructions
FEAT_CCIDX	Yes	Extended cache index
FEAT_SPEv1p1	No	Statistical Profiling Extensions version 1.1
FEAT_DoPD	Yes	Debug over Powerdown
FEAT_PAuth2	Yes	Enhancements to pointer authentication
FEAT_FPAC	Yes	Faulting on pointer authentication instructions *Faulting Pointer Authentication Code* (FPAC)
FEAT_FPACCOMBINE	Yes	Faulting on combined pointer authentication instructions

Table 2-6: Implementation status of the Arm®v8.4-A features in the Cortex-A520 core

Feature	Implemented	Description
FEAT_SEL2	Yes	Secure EL2
FEAT_NV2	No	Enhanced support for nested virtualization
FEAT_S2FWB	Yes	Stage 2 forced write-back
FEAT_DIT	Yes	Data Independent Timing instructions
FEAT_IDST	Yes	ID space trap handling
FEAT_FlagM	Yes	Condition flag manipulation
FEAT_LSE2	Yes	Large System Extensions version 2
FEAT_LRCPC2	Yes	Load-acquire RCpc instructions version 2
FEAT_TLBIOS	Yes	TLB invalidate outer-shared instructions
FEAT_TLBIRANGE	Yes	TLB range invalidate range instructions
FEAT_TTL	Yes	Translation Table Level
FEAT_BBM	Yes	Translation table break before make levels

Feature	Implemented	Description
FEAT_RASv1p1	Yes	*Reliability, Availability, and Serviceability* (RAS) Extension version 1.1 All extensions up to Arm®v9.0-A at full containment capability with *Error Correcting Code* (ECC) configured. See 11. RAS Extension support on page 86 for more information on the implementation of this extension in the core.
FEAT_DoubleFault	Yes	Double Fault Extension
FEAT_Debugv8p4	Yes	Debug relaxations and extensions version 8.4
FEAT_PMUv3p4	Yes	PMU extension version 3.4
FEAT_TRF	Yes	Self hosted Trace Extensions
FEAT_TTST	Yes	Small translation tables
FEAT_AMUv1	Yes	Activity Monitors Extension
FEAT_MPAM	Yes	*Memory Partitioning and Monitoring* (MPAM) For more information on the *Memory System Resource Partitioning and Monitoring* (MPAM) Extension, see the *Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture*.

Table 2-7: Implementation status of the Arm®v8.5-A features in the Cortex-A520 core

Feature	Implemented	Description
FEAT_FlagM2	Yes	Condition flag manipulation version 2
FEAT_FRINTTS	Yes	FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions
FEAT_ExS	No	Disabling context synchronizing exception entry and exit
FEAT_GTG	Yes	Guest translation granule size
FEAT_BTI	Yes	*Branch Target Identification* (BTI)
FEAT_E0PD	Yes	Preventing EL0 access to halves of address maps
FEAT_RNG	No	Random number generator
FEAT_RNG_TRAP	No	Trapping support for RNDR and RNDRRS
FEAT_MTE	Yes	Instruction-only Memory Tagging Extension The Cortex-A520 core always implements the *Memory Tagging Extension* (MTE) and therefore is compliant with the CHI.E protocol. For information on CHI.E commands inferred by MTE, see the *CHI master interface* chapter in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*.
FEAT_MTE2	Yes	Full Memory Tagging Extension
FEAT_MTE3	Configurable These features are enabled by setting the BROADCASTMTE pin to 1.	MTE Asymmetric Fault Handling
FEAT_PMUv3p5	Yes	PMU Extension version 3.5

Table 2-8: Implementation status of the Arm®v8.6-A features in the Cortex-A520 core

Feature	Implemented	Description
FEAT_ECV	Yes	Enhanced counter virtualization
FEAT_FGT	Yes	Fine Grain Traps
FEAT_TWED	No	Delayed trapping of WFE
FEAT_AMUv1p1	No	Activity Monitors Extension version 1.1
FEAT_MPAMv0p1	No	Memory Partitioning and Monitoring version0.1
FEAT_MPAMv1p1	Yes	Memory Partitioning and Monitoring version1.1
FEAT_MTPMU	No	Multi-threaded PMU Extensions

Table 2-9: Implementation status of the Arm®v8.7-A features in the Cortex-A520 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/stores without return
FEAT_LS64_V	No	Support for 64-byte stores with return
FEAT_LS64_ACCDATA	No	Support for 64-byte EL0 stores with return
FEAT_PMUv3p7	Yes	Arm®v8.7-A PMU Extensions See 18.1 Performance monitors events on page 113.
FEAT_RPRES	No	Increased precision of Reciprocal Estimate and Reciprocal Square Root Estimate
FEAT_SPEv1p2	No	Arm®v8.7-A SPE
FEAT_WFxT	Yes	WFE and WFI instructions with timeout See 5.2.1 Wait for Interrupt and Wait for Event on page 48.
FEAT_XS	Yes	XS attribute

Table 2-10: Implementation status of the Arm®v8.8-A features in the Cortex-A520 core

Feature	Implemented	Description
FEAT_CMOW	No	Control for cache maintenance permission
FEAT_Debugv8p8	No	Debug v8.8
FEAT_HBC	No	Hinted conditional branch
FEAT_HPMN0	Yes	Setting of MDCR_EL2.HPMN to zero
FEAT_MOPS	No	Standardization of memory operations
FEAT_NMI	No	Non-maskable Interrupts
FEAT_PMUv3p8	No	Arm®v8.8-A PMU Extensions
FEAT_PMUv3_TH	No	Event counting threshold
FEAT_SPEv1p3	No	Arm®v8.8-A Statistical Profiling Extensions
FEAT_TIDCP1	No	EL0 use of IMPLEMENTATION DEFINED functionality

Table 2-11: Implementation status of the Arm®v9.0-A features in the Cortex-A520 core

Feature	Implemented	Description
FEAT_ETE	Yes	*Embedded Trace Extension* (ETE) See 19. Embedded Trace Extension support on page 143.
FEAT_SVE2	Yes	*Scalable Vector Extension* (SVE) version 2 See 15. Scalable Vector Extensions support on page 99.
FEAT_SVE_AES	Yes	SVE AES instructions
FEAT_SVE_PMULL128	Configurable	SVE PMULL instructions
FEAT_SVE_SHA3		SVE SHA-3 instructions
FEAT_SVE_SM4	Supported as part of the Arm®v8-A Cryptographic Extension	SVE SM4 instructions
FEAT_SVE_BitPerm	Yes	SVE Bit Permute
FEAT_TME	No	*Transactional Memory Extension* (TME)
FEAT_TRBE	Yes	*TRace Buffer Extension* (TRBE) See 20. Trace Buffer Extension support on page 156.

Table 2-12: Implementation status of the Arm®v9.1-A features in the Cortex-A520 core

Feature	Implemented	Description
FEAT_ETEv1p1	Yes	Embedded Trace Extension, version 1.1

Table 2-13: Implementation status of the Arm®v9.2-A features in the Cortex-A520 core

Feature	Implemented	Description
FEAT_BRBE	No	Branch Record Buffer Extensions
FEAT_RME	No	Realm Management Extension
FEAT_ETEv1p2	No	Embedded Trace Extension, version 1.2
FEAT_SME	No	Scalable Matrix Extension
FEAT_SME_FA64	No	Full A64 support in Streaming mode
FEAT_SME_F64F64	No	Double-precision floating-point outer product instructions
FEAT_SME_I16I64	No	16-bit to 64-bit integer widening outer product instructions
FEAT_EBF16	No	Enhanced BFloat16

The following table shows the other standards and specifications that the Cortex-A520 core supports.

Table 2-14: Other standards and specifications supported in the Cortex-A520 core

Standard or specification	Version	Description
FEAT_GICv4p1	GICv4.1	*Generic Interrupt Controller* (GIC) See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information.

Standard or specification	Version	Description
Debug	-	Arm®v9.2-A architecture implemented with Arm®v8.4-A Debug architecture support and Arm®v8.3-A Debug over powerdown support. See the *Arm® Architecture Reference Manual for A-profile architecture* for information on this architecture.
CoreSight	v3.0	See the *Arm® CoreSight™ Architecture Specification v3.0* for more information.

Related information

3.1 Complex Components on page 37

2.5 Test features

The Cortex-A520 core provides test signals that enable the use of both *Automatic Test Pattern Generation* (ATPG) and *Memory Built-In Self Test* (MBIST) to test the core logic and memory arrays.

The Cortex-A520 core includes an ATPG test interface that provides signals to control the *Design for Test* (DFT) features of the core. To prevent problems with DFT implementation, you must carefully consider how you use these signals.

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

For the list of test signals and information on their usage, see the *Design for Test integration guidelines* chapter in the *Arm® Cortex-A520 Core Configuration and Integration Manual*.

For the list of external scan control signals, see the *Design for Test integration guidelines* chapter in the *Arm®DynamIQ™ Shared Unit-120 Configuration and Integration Manual*.


Note

The *Arm® Cortex-A520 Core Configuration and Integration Manual* and *Arm®DynamIQ™ Shared Unit-120 Configuration and Integration Manual* are confidential documents that are available with the appropriate product licenses.

2.6 Design tasks

The Cortex-A520 core is delivered as a synthesizable RTL description in SystemVerilog. Before you can use the Cortex-A520 core, you must implement, integrate, and program it.

A different party can perform each of the following tasks:

Implementation

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

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

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

Integration

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

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

Software programming

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

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

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

See *Functional integration* in the *Arm®DynamIQ™ Shared Unit-120 Configuration and Integration Manual* for signal descriptions.

See *RTL configuration process* in the *Arm® Cortex-A520 Core Configuration and Integration Manual* and in the *Arm®DynamIQ™ Shared Unit-120 Configuration and Integration Manual* for implementation options.

2.7 Product revisions

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

Table 2-15: Product revisions

Revision	Notes
r0p0	First limited access release
r0p1	Added support for FEAT_ECBHB. *Exploitative Control using Branch History Buffer* information between exception levels.
r0p2	Bug fixes

Changes in functionality that have an impact on the documentation also appear in C.1 Revisions on page 812.

3. Technical overview

The components in the Cortex-A520 core are designed to make it a high-efficiency core.

The main blocks include:

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

The Cortex-A520 core interfaces with the *DynamIQ Shared Unit-120* through the CPU bridge.

The Cortex-A520 core implements the Arm®v9.2-A architecture. The Arm®v9.2-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.7-A.

The programmer's model and the architecture features implemented, such as the Generic Timer, are compliant with the standards in 2.4 Supported standards and specifications on page 28.

3.1 Complex Components

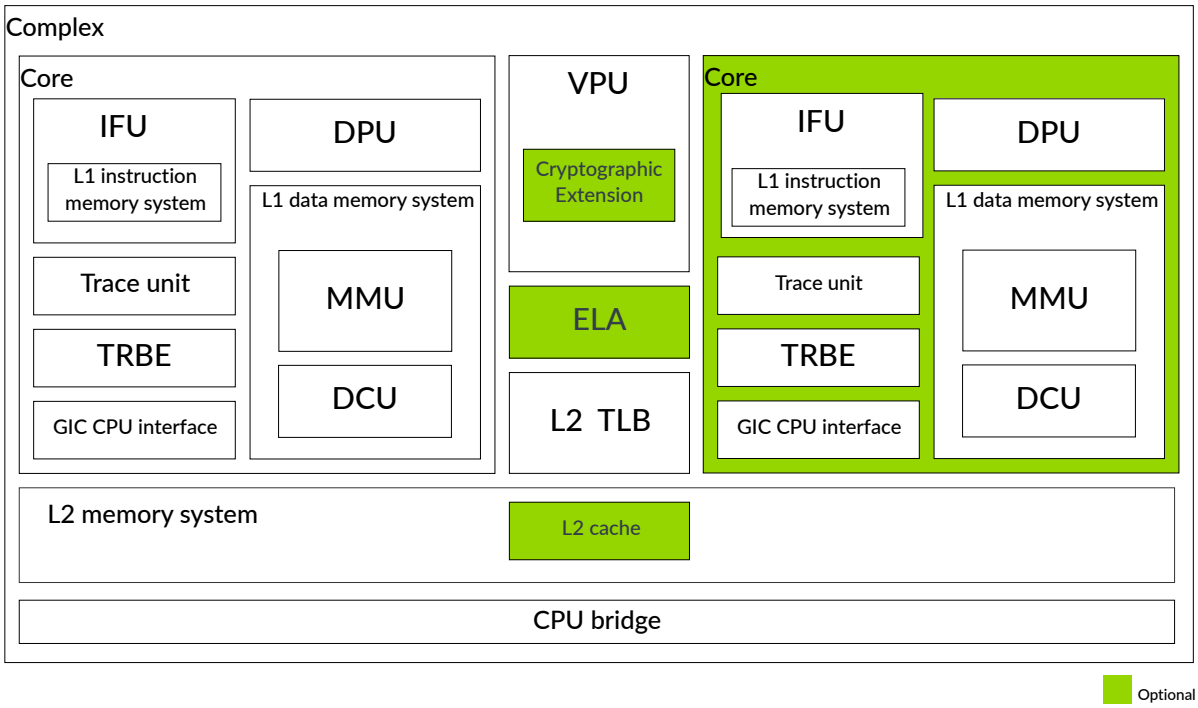
The Cortex-A520 complex includes components that are designed to make it a high-efficiency, low-power, and area-efficient product.

Cortex-A520 cores are always implemented inside a complex.

A Cortex-A520 complex includes a CPU bridge that connects the complex to the *DynamIQ Shared Unit-120*. The DSU-120 connects the complex to an external memory system and to the rest of the *System on Chip* (SoC).

The following figure shows the components within a Cortex-A520 complex.

Figure 3-1: Cortex-A520 complex components



Instruction Fetch Unit

The *Instruction Fetch Unit* (IFU) fetches instructions from the instruction cache or from external memory and uses a dynamic branch predictor to predict the outcome of branches in the instruction stream. It passes the instructions to the *Data Processing Unit* (DPU) for processing.

The L1 instruction memory system includes:

- A fully associative L1 instruction *Translation Lookaside Buffer* (TLB)
- A 32KB or 64KB 4-way set associative L1 instruction cache with 64-byte cache lines

Data Processing Unit

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

Performance Monitoring Unit

The PMU provides either six or 20 performance monitors that can be configured to gather statistics on the operation of each core and the memory system. The information can be used for debug and code profiling.

Activity Monitoring Unit

The Cortex-A520 core includes an AMU, which, like the PMU, counts certain events that are related to the behavior of the core. The AMU implements seven event counters. Activity

monitoring is intended for system management use whereas performance monitoring is aimed at user and debug applications.

The AMU registers are accessible using the System registers or the DSU-120 DynamIQ™ cluster utility bus.

L1 data memory system

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

The L1 data memory system includes:

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

The MMU provides fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes that are held in translation tables. The TLB stores these mappings when translating an address.

Trace Unit and Trace Buffer Extension

The Cortex-A520 core supports a range of debug, test, and trace options including instruction-trace-only trace unit and *TRace Buffer Extension* (TRBE).

The Cortex-A520 core also includes a ROM table that enumerates the debug components within the complex. Debuggers can use the ROM table to determine which CoreSight components are implemented.

All the debug and trace components of the Cortex-A520 core are described in the *Arm® Cortex-A520 Core Configuration and Integration Manual*.

GIC CPU interface

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

Vector Processing Unit

The Cortex-A520 complex includes a *Vector Processing Unit* (VPU) that is shared between the cores of a dual-core complex. Single-core complexes have a dedicated VPU.

The VPU supports Advanced SIMD and floating-point operation. Advanced SIMD is a media and signal processing architecture that adds instructions primarily for audio, video, 3D graphics, and image and speech processing. The floating-point architecture supports half-precision, single-

precision and double-precision floating-point operations. The VPU also supports the *Scalable Vector Extension* (SVE) and SVE2 SIMD instruction sets. SVE and SVE2 complement the Advanced SIMD and floating-point functionality.


Note

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

Cryptographic Extension

The Cryptographic Extension is optional in the Cortex-A520 cores. The Cryptographic Extension adds new instructions to the Advanced SIMD and the SVE instruction sets that accelerate:

- *Advanced Encryption Standard* (AES) encryption and decryption
- The *Secure Hash Algorithm* (SHA) functions SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, and SHA-3
- SM3 hash function and SM4 encryption and decryption
- Finite field arithmetic that is used in algorithms such as Galois/Counter Mode and Elliptic Curve Cryptography


Note

The optional Cryptographic Extension is not included in the base product. Arm supplies the Cryptographic Extension under a separate license to the Cortex-A520 core license.

L2 TLB

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

The TLB entries are global or can include *Address Space Identifiers* (ASIDs) to prevent context switch TLB cleans. They also include *Virtual Machine Identifiers* (VMIDs) to prevent TLB cleans on virtual machine switches by the hypervisor. The Cortex-A520 core can also use the *Common not Private* (CnP) architectural feature that permits cores in a complex to share L2 TLB entries.

L2 memory system

The L2 memory system includes the optional L2 cache. The L2 cache is private to the complex and is 8-way set associative. You can configure the L2 cache size to be 128KB, 192KB, 256KB, 384KB or 512KB. The L2 memory system is connected to the DSU-120 through the CPU bridge.

The L2 cache can be configured to have one or two cache slices. Each slice consists of L2 tag and data RAMs, L2 replacement RAM, L1 duplicate tag RAMs, and associated logic. If two slices are present, most traffic from the cores, the L2 TLB, and from downstream snoops is striped across the slices, based on the value of address bit[6]. This striping increases overall throughput. Accesses to

Device non-reorderable memory and to *Distributed Virtual Memory* (DVM) operations are always handled by slice 0.

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

CPU bridge

In a DynamIQ™ cluster, there is one CPU bridge between each Cortex-A520 complex and the DSU-120.

The CPU bridge controls buffering and synchronization between the complex and the DSU-120.

By default, the CPU bridge is asynchronous to permit different *Power Performance and Area* (PPA) implementation points for each complex and the DSU-120. When the CPU bridge runs asynchronously, the core and the DynamIQ™ cluster can run at different frequencies. You can, however, configure the CPU bridge to run synchronously with the memory bus interface without affecting the other asynchronous interfaces such as debug and trace.

See *RTL configuration process* in the *Arm®DynamIQ™ Shared Unit-120 Configuration and Integration Manual* for more information.

Related information

6. Memory management on page 62
7. L1 instruction memory system on page 70
8. L1 data memory system on page 73
9. L2 memory system on page 80
10. Direct access to internal memory on page 83
13. GIC CPU interface on page 95
14. Advanced SIMD and floating-point support on page 98
18. Performance Monitors Extension support on page 113
19. Embedded Trace Extension support on page 143

3.2 Interfaces

The *DynamIQ Shared Unit-120* manages all Cortex-A520 core external interfaces to the *System on Chip* (SoC).

See the *Technical overview* chapter in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for detailed information on these interfaces.

3.3 Programmer's model

The Cortex-A520 core implements the Arm®v9.2-A architecture. The Arm®v9.2-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.7-A. The Cortex-A520 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

2.4 Supported standards and specifications on page 28

4. Clocks and resets

To provide dynamic power savings, the Cortex-A520 core supports hierarchical clock gating. It also supports Warm and Cold resets.

Each Cortex-A520 complex has a single clock domain and receives a single clock input. This clock input is gated by an architectural clock gate in the CPU bridge. There is one architectural clock gate per core in the complex, and one for the shared logic. If the complex is configured with an asynchronous bridge, the clock input is COMPLEXCLK<n>, where n indicates the number of the complex within the DSU-120 DynamIQ™ cluster. If the complex is not configured with an asynchronous bridge, the clock input is SCLK clock signal.

In addition, the Cortex-A520 core implements extensive clock gating that includes:

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

The Cortex-A520 core receives the following reset signals from the *DynamIQ Shared Unit-120* 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 unit logic
  - *Reliability, Availability, and Serviceability* (RAS) logic
- A Cold reset for the logic in the complex, including the debug logic, trace logic, and RAS logic.

For a complete description of the clock gating and reset scheme of the complex, see the following sections in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*:

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

5. Power management

The Cortex-A520 core provides mechanisms to control both dynamic and static power dissipation.

The dynamic power management includes the following features:

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

The static power management includes the following features:

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

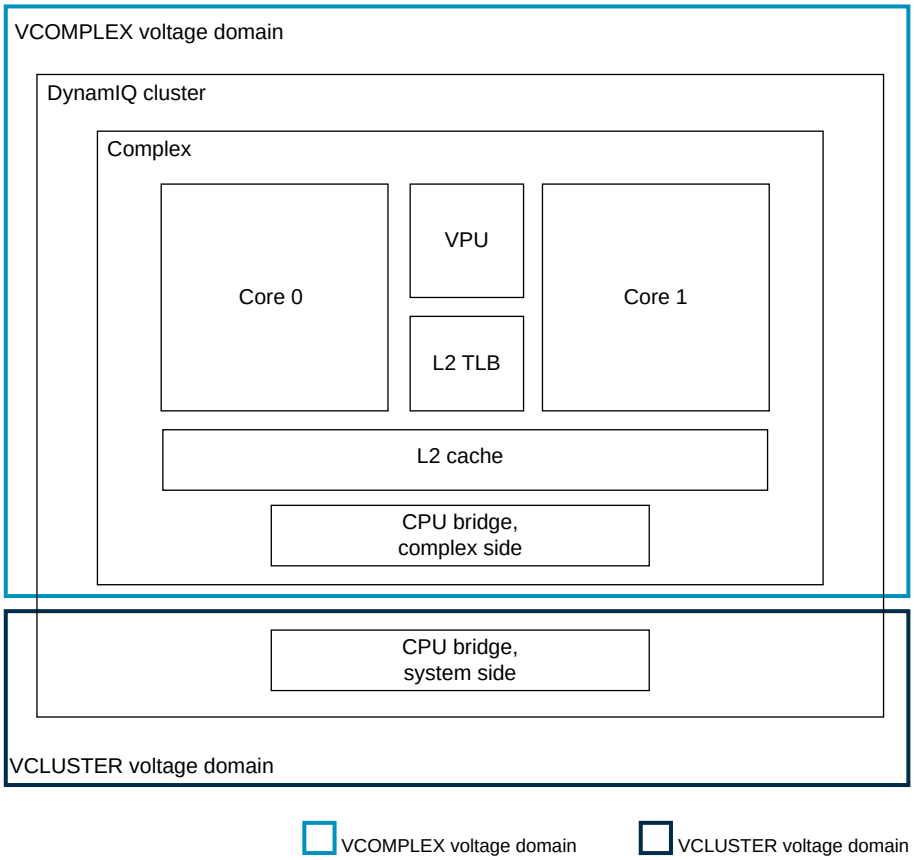
5.1 Voltage and power domains

The *DynamIQ Shared Unit-120 Power Policy Units* (PPUs) control power management for the Cortex-A520 core.

A Cortex-A520 complex supports separate gated power domains for the complex, for each core inside the complex, and for the *Vector Processing Unit* (VPU). It also supports a dedicated voltage domain for each complex, and a voltage domain for the DSU-120 DynamIQ™ cluster.

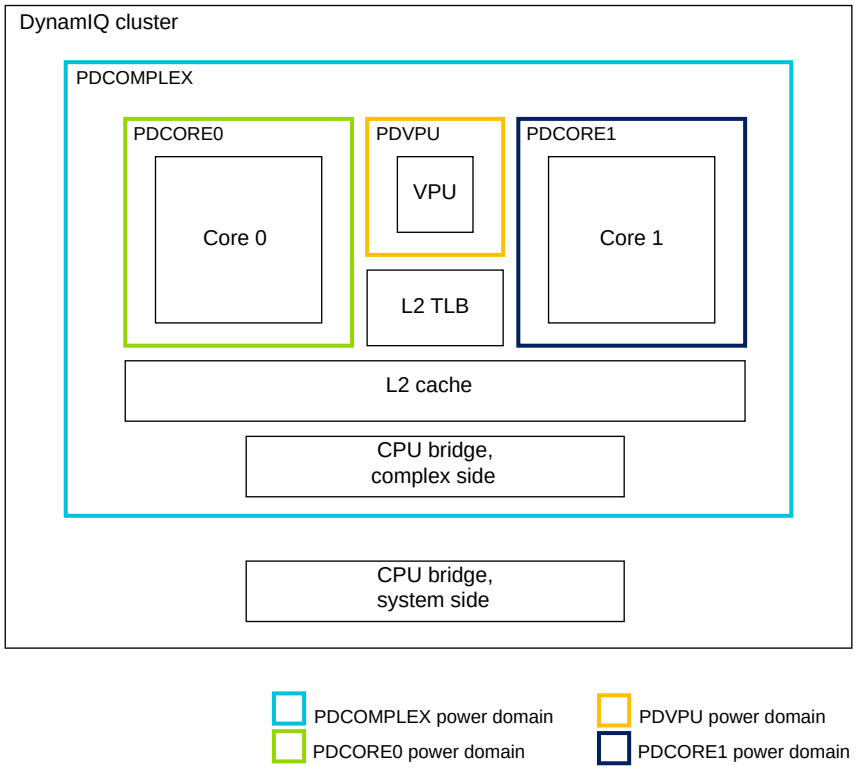
The following figure shows the voltage domains for a Cortex-A520 configuration with a dual-core complex.

Figure 5-1: Cortex-A520 voltage domains, dual-core complex



The following figure shows the power domains for an example Cortex-A520 configuration with a dual-core complex.

Figure 5-2: Cortex-A520 power domains, dual-core complex



A Cortex-A520 complex is instantiated within a DynamIQ™ cluster. Within the complex, the system side of the CPU bridge is within the cluster voltage domain, VCLUSTER. From the perspective of the complex, the system side of the CPU bridge is always on.

The remainder of the complex logic is in a separate VCOMPLEX voltage domain and PDCOMPLEX power domain. Within the PDCOMPLEX power domain, each core is in the PDCORE<n> power domain, where n is the core instance number.

The VPU is in the PDVPU power domain. The rest of the shared logic, consisting of the L2 *Translation Lookaside Buffer* (TLB), the L2 cache, and the complex side of the CPU bridge is in the PDCOMPLEX power domain.

PDCORE<n> is a gated power domain that can support retention. See *The DynamIQ Shared Unit* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for more information about instance numbering.

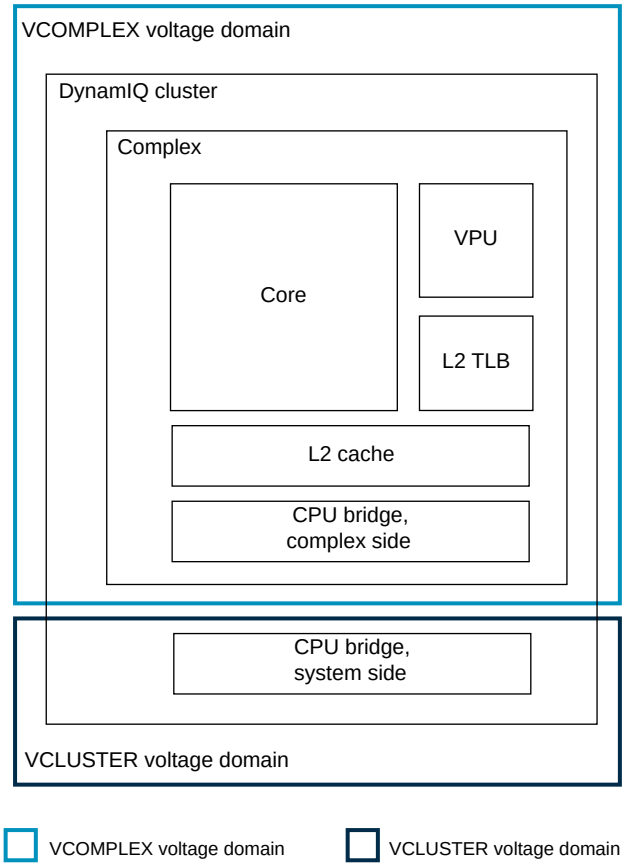
The VCOMPLEX voltage domain operates within a single clock domain, COMPLEXCLK. The CPU bridge contains high-level clock gates and generates gated clocks corresponding to each gated power domain. Also, the clock to the VPU is gated when the VPU is idle.

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

The CPU bridge logic within the VCLUSTER voltage domain operates within multiple clock domains. See *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for more information.

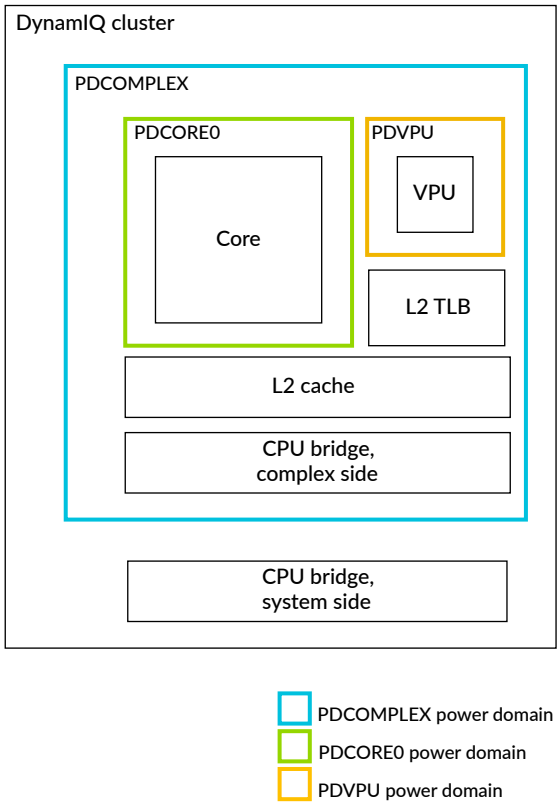
The following figure shows the voltage domains for a Cortex-A520 configuration with a single-core complex.

Figure 5-3: Cortex-A520 voltage domains, single-core complex



The following figure shows the power domains for a Cortex-A520 configuration with a single-core complex.

Figure 5-4: Cortex-A520 power domains, single-core complex



For a single-core complex, the voltage and power domains are similar to those for a dual-core complex.

Within the PDCOMPLEX power domain, the single core is in PDCORE0, a gated power domain that can support retention. The core has its own dedicated logic, including a VPU within its own PDVPU power domain. The L2 TLB, the L2 cache, and the CPU bridge, complex side, is in the PDCOMPLEX power domain.

5.2 Architectural clock gating modes

The `WFI`, `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.

5.2.1 Wait for Interrupt and Wait for Event

Wait for Interrupt (WFI) and *Wait for Event* (WFE) are features that put a core within a Cortex-A520 complex in a low-power state by disabling most of the core clocks, while keeping the core powered

up. When the core is in WFI or WFE state, the input clock is gated externally to the core at the CPU bridge.

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.

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

5.2.2 Low-power state behavior considerations

You must consider how certain events affect the *Wait for Interrupt* (WFI) and *Wait for Event* (WFE) low-power state behavior of the Cortex-A520 complex.

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 APB interface
- A system snoop request that must be serviced by the core L1 data cache
- A cache or *Translation Lookaside Buffer* (TLB) maintenance operation that must be serviced by the core L1 instruction cache, L1 data cache, L2 cache, or TLB
- Any access from the other core in the complex that must be serviced by the L1 data cache


Note

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

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

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

5.3 Power control

The *DynamIQ Shared Unit-120 Power Policy Units* (PPUs) control all core and cluster power mode transitions.

Each core within a Cortex-A520 complex has an individual PPU for controlling its own core power domain. For example, there is a PPU for PDCORE0 and a PPU for PDCORE1.

In addition, there is a PPU for the cluster.

The PPUs decide and request any change in power mode. The targeted core within the Cortex-A520 complex then performs any actions necessary to reach the requested power mode. For example, the core might gate clocks, clean caches, or disable coherency before it accepts the request.

For more information about the PPUs for the cluster and the cores, see the following sections in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*:

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

Related information

A.2.1 IMP_CPUPPMCR_EL3, Global PPM Configuration Register on page 236
A.5.25 IMP_CPUMPMMCR_EL3, Global MPMM Configuration Register on page 325
B.1.1 CPUPPMCR, Global PPM Configuration Register on page 460
B.1.2 CPUMPMMCR, Global MPMM Configuration Register on page 462

5.4 Core power modes

Each core in a Cortex-A520 complex, as well as the shared logic, has a defined set of power modes and corresponding legal transitions between these power modes. The power mode of each core can be independent of other cores in a complex or DSU-120 DynamIQ™ cluster.

The *Power Policy Unit* (PPU) of a core manages at the cluster level the transitions between the power modes for that core. See *Power management* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for more information.

The following table shows the supported Cortex-A520 core power modes.

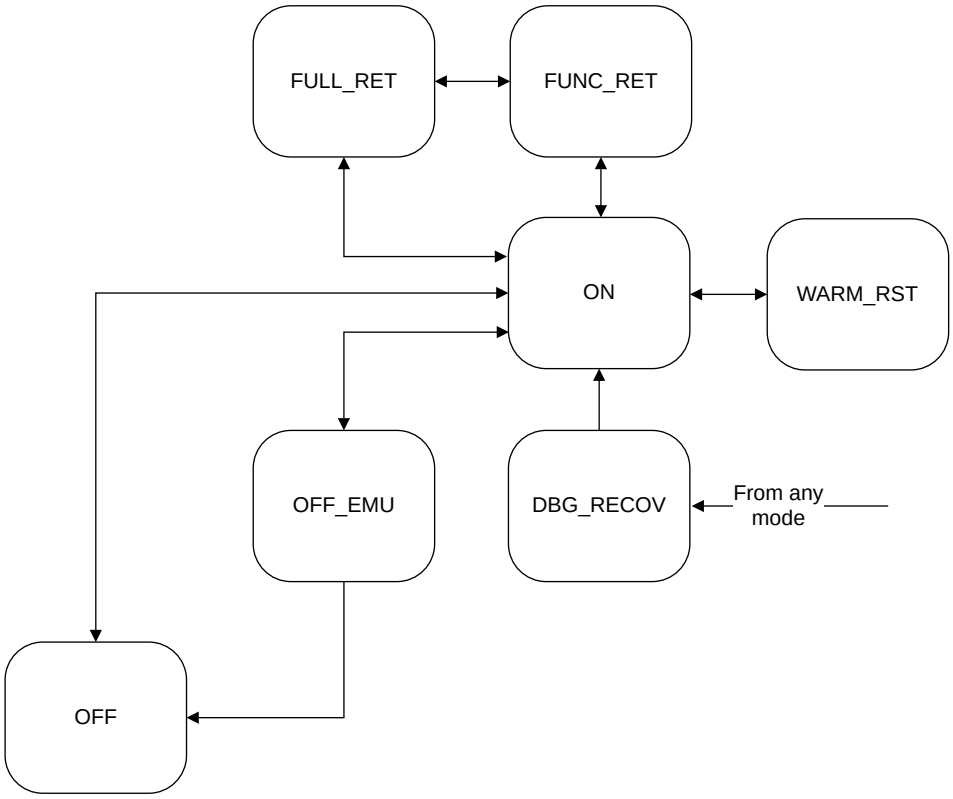
Power modes that are not shown in the following table are not supported and must not occur. Deviating from the legal power modes can lead to **UNPREDICTABLE** results. You must comply with the dynamic power management, and powerup and powerdown sequences described in 5.7 Cortex-A520 core powerup and powerdown sequence on page 58.

Table 5-1: Cortex-A520 core power modes

Power mode	Short name	Power state
On	ON	The core is powered up and active.
Functional retention	FUNC_RET	The core is fully powered and operational, but the *Vector Processing Unit* (VPU) is idle.
Full retention	FULL_RET	The core is in retention. 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	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.
Debug recovery	DBG_RECOV	The RAM and logic are powered up. This mode is for applying a Warm reset to the DSU-120 DynamIQ™ cluster, while preserving memory and *Reliability, Availability, and Serviceability* (RAS) registers for debug purposes. Both cache and RAS state are preserved when transitioning from DBG_RECOV to ON. **Caution:** This mode must not be used during normal system operation.
Warm reset	WARM_RST	A Warm reset resets all state except for the debug logic, the 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 Cortex-A520 core power domain and the legal transitions between them.

Figure 5-5: Permitted Cortex-A520 core power mode transitions



Related information

5.2 Architectural clock gating modes on page 48
5.2.1 Wait for Interrupt and Wait for Event on page 48
5.4.5 Full retention mode on page 53

5.4.1 On mode

In the On power mode, the Cortex-A520 core is on and fully operational.

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

5.4.2 Off mode

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

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


Note

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

An attempted debug access or utility bus access to the core when the core domain is off returns an error response on the utility bus, indicating that the core is not available. See *Utility bus* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for more information.

5.4.3 Emulated off mode

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

5.4.4 Functional retention mode

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

In Functional retention mode, the core is fully powered and operational, but the *Vector Processing Unit* (VPU) is off. The VPU can enter this mode when it is idle and the retention timer has expired. Software can enable or disable this mode and set the length of the retention timer.

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

Related information

A.1.13 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 201

5.4.5 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 retention timer has expired. For more information on setting the retention timer, see A.1.13 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 201.
- The core is in *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) low-power state.

- The core clock is not temporarily enabled for any of the following reasons:
  - L1 snoops or L2 snoops
  - Cache or *Translation Lookaside Buffer* (TLB) maintenance operations
  - Debug or *Generic Interrupt Controller* (GIC) access

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 on the debug *Advanced Peripheral Bus* (APB)
  - GIC access

Related information

5.2.1 Wait for Interrupt and Wait for Event on page 48

5.4.6 Debug recovery mode

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

By default, the core invalidates its caches when it transitions from Off to On mode. Using Debug recovery mode allows the L1 cache and L2 cache contents that were present before the reset to be observable after the reset. 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. 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 *DynamIQ Shared Unit-120 Power Policy Units* (PPUs).

For more information on the DSU-120 *Power Policy Units* (PPUs), see *The Power Policy Unit* in the *Arm® DynamIQ™ Shared Unit-120 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 DSU-120 DynamIQ™

cluster, or the wider system are **UNPREDICTABLE**, and a wider system reset might be required. In particular, any outstanding memory system transactions at the time of the reset might complete after the reset. The core is not expecting these transactions to complete after a reset, and might cause a system deadlock.

If the system sends a snoop to the DSU-120 DynamIQ™ cluster during Debug recovery mode, depending on the cluster state:

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

5.4.7 Warm reset mode

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

A Warm reset is applied to the Cortex-A520 core when the core *Power Policy Unit* (PPU) in the *DynamIQ Shared Unit-120* is programmed for WARM_RST mode.

Warm reset 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.

The Cortex-A520 core also implements the Arm®v8-A Reset Management Register, RMR_EL3. When the core runs in EL3 and is in *Wait for Interrupt* (WFI), it requests a Warm reset of the core if you set the RMR_EL3.RR bit to 1.

See the *Arm® Architecture Reference Manual for A-profile architecture* for more information about RMR_EL3.

5.5 Complex power modes

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

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

The *Power Control State Machine* (PCSM) interface is an external interface for controlling low-level technology-specific power switch and retention controls.

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

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

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

PPU mode		PCSM channel		
Core0	Core1	Core0	Core1	Shared logic
Off	Off	OFF	OFF	OFF


Note

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

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

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

In a dual-core complex, where a single core is being powered down, the shared logic might need to be kept powered on. The powerdown sequence must account for this possibility. Both the L2 cache and the VPU are shared in a dual-core complex. Therefore, when a core in a Cortex-A520 complex is being powered down:

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

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

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

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

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

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

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

Related information

5.3 Power control on page 50

5.6 Performance and power management

The Cortex-A520 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 feature that Cortex-A520 core implements is:

- *Maximum Power Mitigation Mechanism* (MPMM)

5.6.1 Maximum Power Mitigation Mechanism

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

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

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

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

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

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

Related information

A.2.1 IMP_CPUPPMCR_EL3, Global PPM Configuration Register on page 236
A.5.25 IMP_CPUMPMMCR_EL3, Global MPMM Configuration Register on page 325
B.1.1 CPUPPMCR, Global PPM Configuration Register on page 460
B.1.2 CPUMPMMCR, Global MPMM Configuration Register on page 462

5.7 Cortex-A520 core powerup and powerdown sequence

There is no specific sequence to power up the Cortex-A520 core. There are no software steps required to bring a core into coherence after reset. For powerdown, the Cortex-A520 core uses a specific sequence.

To power down the Cortex-A520 core:

1. If necessary, save the state of the core to system memory, so that it can be restored during the core powerup.
2. Disable interrupts to the core.
   a. Disable the interrupt enable bits in the ICC_IGRPEN0_EL1 and ICC_IGRPEN1_EL1 registers.
   b. Set the GIC Distributor wake up request for the core using the GICR_WAKER register.
   c. Read the GICR_WAKER register to confirm that the ChildrenAsleep bit indicates that the interface is idle.
3. 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. See Managing RAS fault and error interrupts during the core powerdown procedure.
4. Set the IMP_CPUPWRCTLR_EL1.CORE_PWRDN_EN bit to 1 to indicate to the power controller that a powerdown is requested.
5. Execute an `ISB` instruction.
6. Execute a `WFI` instruction.

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

- Disables and cleans the core cache
- Removes the core from coherency

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

Related information

A.1.13 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 201

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

The `WFI` instruction is the point of no return for powering down the core. For this reason, the power management architecture does not permit interrupting the core software after this `WFI` instruction is executed.

Therefore, the core software cannot be interrupted to manage any *Reliability, Availability, and Serviceability* (RAS) fault or error which is either:

- Detected before the core powerdown procedure executes the `WFI` instruction and is not cleared
- Detected after the core powerdown procedure executes the `WFI` instruction.

Any RAS fault or error interrupt output from the core that is active prevents the core from powering down. This means that:

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

Therefore, the status of the RAS fault and error interrupts must be managed as part of the core powerdown sequence to prevent this situation from occurring.

The two general options for managing RAS fault and error interrupts during the core powerdown procedure are:

- Disable the generation of RAS fault and error interrupts using the ERxCTLR_EL1 registers and clear any current RAS fault or error interrupts before the core powerdown procedure executes the `WFI` instruction.
- Reroute the RAS fault or error interrupts to a separate system error management device as part of the powerdown procedure. This device, such as a System Control Processor, is responsible for resetting the system if a fault or error is signaled. However, this approach is only possible if the system has been designed to allow the RAS interrupt outputs to be rerouted to another component.

If all the RAS fault and error interrupt outputs are disabled before the core powerdown procedure but the error detection and correction response are still enabled, then:

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

In some systems it might be preferable to disable the generation of RAS fault and error interrupts for correctable and deferrable errors but to enable the error interrupt for uncorrectable errors as follows: ERxCTLR_EL1.CFI = 0, ERxCTLR_EL1.FI = 0, and ERxCTLR_EL1.UI = 1. Using this approach, the core error interrupt output must be rerouted to the system error manager before executing the `WFI` instruction in the core powerdown procedure. If an uncorrectable error occurs during the powerdown, the core remains powered ON but the software is inactive. The system error manager is then responsible for resetting the entire cluster and the wider system that is interacting with the core and cluster. To use this approach, the system must permit the core RAS error interrupt to be rerouted to the system error manager. However, the system error manager is unable to identify where the uncorrectable error occurred within the core because the core RAS registers are only accessible to software running on the core.

5.8 Debug over powerdown

The Cortex-A520 core supports debug over powerdown, which allows a debugger to retain its connection with the core even when powered down. This behavior enables debug to continue through powerdown scenarios, rather than having to re-establish a connection each time the core is powered up.

The debug over powerdown logic is part of the DebugBlock in the *DynamIQ Shared Unit-120*. The DebugBlock is external to the DSU-120 DynamIQ™ cluster and must remain powered on during the debug over powerdown process.

See *Debug* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for more information.

6. 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 Cortex-A520 core internal registers and translation tables. The MMU also controls memory access permissions, memory ordering, and cache policies for each region of memory.

An address translation from an input address to an output address is described as a stage of address translation. The Cortex-A520 core can perform:

- Stage 1 translations that translate an input *Virtual Address* (VA) to an output *Physical Address* (PA) or *Intermediate Physical Address* (IPA).
- Stage 2 translations that translate an input IPA to an output PA.
- Combined stage 1 and stage 2 translations that translate an input VA to an IPA, and then translate that IPA to an output PA. The Cortex-A520 core performs translation table walks for each stage of the translation.

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

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

The Cortex-A520 core supports the *Common not Private* (CnP) feature. CnP is an architectural feature that permits cores in a complex to share translation tables. When CnP is enabled and in use, all cores in a complex can share L2 TLB entries and make better use of the TLB. Without it, each core in a complex might cache the same translation, reducing the effective size of the TLB.

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

6.1 Memory Management Unit components

The Cortex-A520 *Memory Management Unit* (MMU) includes several *Translation Lookaside Buffers* (TLBs) and a translation table prefetcher.

A TLB is a cache of recently executed page translations within the MMU. The Cortex-A520 core implements a two-level TLB structure. The TLB stores all translation table sizes and is responsible for breaking these down into smaller tables when required for the L1 data or instruction TLB.

The following table describes the MMU components.

Table 6-1: MMU components

Component	Description
L1 instruction TLB	• 16 entries • Fully associative • Located in the L1 instruction memory block • TLB hits return the *Physical Address* (PA) to the instruction cache
L1 data TLB	• 16 entries • Fully associative • Located in the L1 data memory block • TLB hits return the PA to the data cache
L2 TLB	• 8-way set associative • A main block that is located within a complex • Shared between the cores of a dual-core complex • Supports dirty bit update, that is, hardware update of access flag and access permissions • Provides translations for instruction side, data side, trace and profiling accesses, and address translation operations
TLB prefetcher	• Prefetches descriptors into the L2 cache, and translations into the L2 TLB • Can be disabled in the IMP_CMPXECTLR_EL1 register

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

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

Related information

A.1.12 IMP_CMPXECTLR_EL1, Complex Extended Control Register on page 196

6.2 Translation Lookaside Buffer match process

The Armv8-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[48:N], where N is $\log_2$ of the block size for that translation that is stored in the TLB entry, matches the requested address.
- Entry translation regime matches the current translation regime.
- The *Address Space IDentifier* (ASID) matches the current ASID held in the TTBR0_ELx or TTBR1_ELx register associated with the target translation regime, or the entry is marked global.
- The *Virtual Machine IDentifier* (VMID) matches the current VMID held in the VTTBR_EL2 register.

The ASID 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.

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

- A core that has CnP disabled cannot use a TLB mapping that is marked as common.
- A core that has CnP enabled cannot use a TLB mapping that is marked as private, even if the mapping was allocated by that core.


Note

A core that has CnP enabled is one where the corresponding TTBR<n>_ELx.CnP field for the core is set to 1. For the Secure EL1 and EL0 and Non-secure EL1 and EL0 translation regimes where EL2 is enabled, CnP is enabled when VTTBR_EL2.CnP is set to 1.

6.3 Translation table walks

When the Cortex-A520 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 Cortex-A520 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.

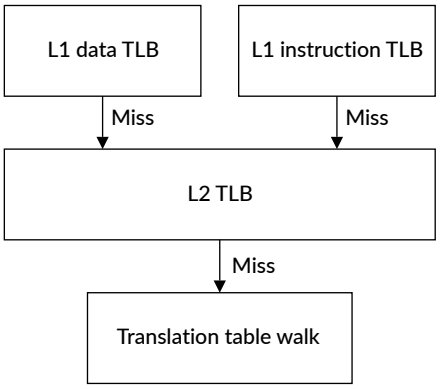
Address translation is performed only when the MMU is enabled. It can also be disabled for a particular translation base register, in which case the MMU returns a Translation Fault.

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

The following figure shows the TLB lookup process.

Figure 6-1: Translation table walks



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

Related information

7. L1 instruction memory system on page 70
8. L1 data memory system on page 73
9. L2 memory system on page 80

6.4 Hardware management of the Access flag and dirty state

The Cortex-A520 core includes the option to perform hardware updates to the translation tables.

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

The Cortex-A520 core supports hardware updates to the Access flag and to dirty state only when the translation tables are held in Inner Write-Back and Outer Write-Back Normal memory regions. If software requests a hardware update in a region that is not Inner Write-Back or Outer Write-Back Normal memory, then the Cortex-A520 core returns an abort with the following encoding:

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

6.5 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
- Data accesses that result from load operations to Normal memory
- Load operations to Device memory, including operations that have acquire semantics


Note

The address captured in the *Fault Address Register* (FAR) is the target address of the instruction that generated the synchronous external abort.

External aborts are reported asynchronously when they occur during:

- Store operations to any memory type
- Cache maintenance, TLB invalidate, and instruction cache invalidate operations

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 VA address space is defined by:

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

The Cortex-A520 core treats such a block as not causing a translation fault and disregards the value of the contiguous bit.

Conflict aborts

The Cortex-A520 core does not generate conflict aborts.

6.6 Memory behavior and supported memory types

The Cortex-A520 core supports memory types defined in the Arm®v8-A architecture.

Device memory types have the following attributes:

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

R – Reordering
: The capability to reorder transactions

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


Note

In the following table, the n prefix means the capability is not allowed.

The following table shows the Device memory types that the Cortex-A520 core supports.

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

Memory type	Description
Device-GRE	Device Gathering, Reordering, Early Write Acknowledgement. Device-GRE is similar to Normal Non-cacheable, but does not permit Speculative accesses.
Device-nGRE	Device non-Gathering, Reordering, Early Write Acknowledgement. Transactions might be reordered within the L3 memory system, or in the system interconnect. The use of barriers is required to order accesses to Device-nGRE memory.
Device-nGnRE	Device non-Gathering, non-Reordering, Early Write Acknowledgement. Device-nGnRE is equivalent to the Device memory type in earlier versions of the architecture.
Device-nGnRnE	Device non-Gathering, non-Reordering, No Early Write Acknowledgement. Device-nGnRnE is treated the same as nGnRE inside the Cortex-A520 core, but reported differently on the bus interface.

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

Write-Through

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

Mixed Inner and Outer Cacheability

Only memory that is marked as Inner and Outer Write-Back can be cached on the data side and make coherency requests. This rule applies to the memory type only, and not to the allocation hints. All caches within the DSU-120 DynamIQ™ cluster are treated as being part of the Inner Cacheability domain.

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

6.7 Page-based hardware attributes

The architecture defines *Page-Based Hardware Attributes* (PBHA) as an optional **IMPLEMENTATION DEFINED** feature. This section describes how the Cortex-A520 core implements PBHA.

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

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

For memory accesses caused by a translation table walk, the 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.

When stage 1 and stage 2 are enabled:

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

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

Mismatched aliases

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

7. L1 instruction memory system

The Cortex-A520 core L1 instruction memory system fetches instructions and predicts branches. It is part of the *Instruction Fetch Unit* (IFU), which includes a dynamic branch predictor. It includes the L1 instruction cache and the L1 instruction *Translation Lookaside Buffer* (TLB).

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

The following table shows the L1 instruction memory system features.

Table 7-1: L1 instruction memory system features

Feature	Description
L1 instruction cache	32KB or 64KB
	4-way set associative
	Virtually-indexed, physically-tagged (VIPT) behaving as *physically-indexed, physically-tagged* (PIPT)
	Single Error Detect (SED) parity cache protection
Cache line length	64 bytes
Cache policy	Dynamic based cache replacement policy


Note

The L1 instruction TLB also resides in the L1 instruction memory system. However, it is part of the *Memory Management Unit* (MMU) and is described in 6. Memory management on page 62.

7.1 L1 instruction cache behavior

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

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

When the instruction Cacheability is disabled, all instruction fetches to Cacheable memory are treated as if they were Non-cacheable. All instruction fetches will not get allocated into instruction cache.

When the instruction Cacheability is enabled, lines might still be allocated into the instruction cache even if the memory is marked as Non-cacheable.

These behaviors mean that instruction fetches might not be coherent with caches in other cores, and software must account for this possibility.

Related information

5.4.6 Debug recovery mode on page 54

7.2 L1 instruction cache Speculative memory accesses

Instruction fetches are Speculative and there can be several unresolved branches in the pipeline. A branch instruction or exception in the code stream can cause a pipeline flush, discarding the currently fetched instructions.

On instruction fetches, pages with Device memory type attributes are treated as Non-Cacheable Normal Memory. 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 a speculative instruction fetches miss in the L1 instruction cache, they can still look into L2 Cache if it is enabled.
See the *Arm® Architecture Reference Manual for A-profile architecture* for more information.

7.3 Program flow prediction

The Cortex-A520 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 conditional branch predictor
- An indirect branch predictor
- Dynamic branch predictor history
- The return stack, a stack of nested subroutine return addresses
- A cache that holds the branch target address of previously taken branches

Predicted and non-predicted instructions

Program flow prediction hardware predicts 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`

8. L1 data memory system

The Cortex-A520 core L1 data memory system is responsible for executing load and store instructions and specific instructions like atomics, cache maintenance operations, and memory tagging instructions. The L1 data memory system includes the L1 data cache and the L1 data *Translation Lookaside Buffer* (TLB).

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

The following table shows the L1 data memory system features.

Table 8-1: L1 data memory system features

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

8.1 L1 data cache behavior

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

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

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

There is no operation to invalidate the entire data cache. If software requires this function, then it must be constructed by iterating over the cache geometry and executing a series of individual invalidates by set/way instructions. The `DC CSW` and `DC ISW` instructions perform both a clean and invalidate of the target set/way. The values of HCR_EL2.SWIO have no effect. See the *Arm® Architecture Reference Manual for A-profile architecture* for more information about `DC CISW` and HCR_EL2.

Data Cacheability disabled behavior

If the data Cacheability is disabled, then:

- Load and store instructions do not access any of the L1 data, L2, or the L3 caches.
- Data cache maintenance operations continue to execute normally.
- Snoop requests continue to access the L1 data, L2, and L3 caches.
- All load and store instructions to cacheable memory are treated as Non-cacheable.

It is not possible to disable data Cacheability for individual levels of cache. When data Cacheability for a core is disabled, other cores can still make accesses and cause allocations to L2 or L3 caches, as can Cacheable instruction fetches.

To maintain data coherency between multiple cores, the Cortex-A520 core uses the *Modified Exclusive Shared Invalid* (MESI) protocol.

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

5.4.6 Debug recovery mode on page 54

8.2 Write streaming mode

The Cortex-A520 core supports write streaming mode, sometimes referred to as read allocate mode, both for the L1 and the L2 cache.

A cache line is allocated to the L1 or L2 cache on either a read miss or a write miss. However, writing large blocks of data can pollute the cache with unnecessary data. It can also waste power and performance when a linefill is performed only to discard the linefill data, because the entire line gets overwritten by subsequent writes (for example using `memset()` or `memcpy()`). 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 detects when the core has written a sequence of full cache lines. 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 look up in the cache, but if they miss, then they write out to the L2 or L3 cache rather than starting a linefill.


Note

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

The write streaming mode remains enabled until either:

- It detects a cacheable write burst that is not a full cache line.
- There is a load operation from the same line that is being written to the L2 or the L3 cache.

When a Cortex-A520 core has switched to write streaming mode, the memory system continues to monitor the bus traffic. It signals to the L2 or L3 cache to go into write streaming mode when it observes a further number of full cache line writes.

The write streaming threshold defines the number of consecutive cache lines that are fully written without being read before store operations stop causing cache allocations. You can configure the write streaming threshold for each cache (L1, L2, and L3) and for the caches outside the cluster by writing the register A.1.11 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 190.

You can configure the write streaming threshold for each cache:

- IMP_CPUECTLR_EL1.L1WSCTL configures the L1 write streaming mode threshold.
- IMP_CPUECTLR_EL1.L2WSCTL configures the L2 write streaming mode threshold.
- IMP_CPUECTLR_EL1.L3WSCTL configures the L3 write streaming mode threshold.
- IMP_CPUECTLR_EL1.L4WSCTL configures the system cache write streaming mode threshold.

8.3 Memory system implementation

The Cortex-A520 core supports a single limited order range that includes the entire memory space. It also has specific behavior for transient memory regions.

Atomic instruction implementation in the L1 data memory system

The Cortex-A520 core supports the atomic instructions added in the Arm®v8.1-A architecture. Atomic instructions to Cacheable memory can be performed either as near atomic or far atomic

instructions. Whether a near or far atomic instruction is used depends on the L1 data cache hit and miss information and on the type of operation. Atomic instruction execution location is as follows:

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

Use IMP_CPUECTLR_EL1.ATOM to configure atomic instruction handling. See the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for more information about atomic instructions.

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

- The interconnect supports far atomics.
- The master interface is configured as CHI.
- The operation misses everywhere within the DSU-120 DynamIQ™ cluster.

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

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

The Cortex-A520 core supports atomics to Device or Non-cacheable memory, however this support relies on the interconnect also supporting atomics. If this type of atomic instruction is executed when the interconnect does not support them, it results in an asynchronous Data Abort.

Transient memory region

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

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

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

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

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

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

Non-temporal loads

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

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


Note

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

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

Related information

A.1.11 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 190

8.4 Internal exclusive monitor

The Cortex-A520 core includes an internal exclusive monitor with a 2-state, open and exclusive state machine that manages Load-Exclusive and Store-Exclusive instructions and Clear-Exclusive instructions.

You can use these instructions to construct semaphores, ensuring synchronization between different processes running on the core. Semaphores can also ensure synchronization between different cores that are using the same coherent memory locations for the semaphore.

A Load-Exclusive instruction tags a small block of memory for exclusive access. The CTR_EL0 register defines the size of the tagged blocks as 16 words, one cache line.


Note

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

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

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

Treatment of intervening store operations

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

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

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

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

Exclusive monitor

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

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

Related information

A.5.23 CTR_EL0, Cache Type Register on page 321

8.5 Data prefetching

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

Preload instructions

For cases that cannot be handled efficiently by data prefetchers, the Cortex-A520 core supports the AArch64 prefetch memory instructions, `PRFM`.

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

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

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

Hardware data prefetcher

The Cortex-A520 core has a data prefetch mechanism that looks for cache line fetches with regular or repetitive patterns of data. The core includes multiple data prefetchers. If a data prefetcher detects a pattern, it signals to the memory system that memory accesses from a specified address are likely to occur soon. The memory system responds by starting new linefills to fetch the predicted addresses ahead of the demand loads. These linefills can be in the L1 data cache, the L2 cache, or the L3 cache, depending on which cache the hardware selects.

Prefetch streams end under any of the following circumstances:

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

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

Data cache zero

In the Cortex-A520 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*.

9. L2 memory system

The Cortex-A520 L2 memory system connects the Cortex-A520 core to the *DynamIQ Shared Unit-120* L3 memory system. It includes an optional unified L2 cache that is private to a complex.

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

For some cores, you can implement the DSU-120 to use the Direct connect feature to connect to the core. However, the Cortex-A520 core does not support Direct connect.

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

- Handles coherent and non-coherent operations from cores and from associated L1 evictions.
- Handles snoop operations from other cores in the DSU-120 DynamIQ™ cluster and from other *Processing Elements* (PEs) in the system, in accordance with the *AMBA® 5 CHI Architecture Specification*.
- Handles instruction cache, *Translation Lookaside Buffer* (TLB), and predictor maintenance operations as *Distributed Virtual Memory* (DVM) messages, including broadcast operations within the complex.

The following table shows the L2 memory system features.

Table 9-1: L2 memory system features

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

9.1 Optional integrated L2 cache

You can implement the Cortex-A520 core with or without an L2 cache.

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

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


Note

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

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

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

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

Related information

5.4.6 Debug recovery mode on page 54
8.2 Write streaming mode on page 74
A.1.11 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 190

9.2 Support for memory types

The Cortex-A520 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. All other memory types are not cached.

Allocation hint

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

The standard CHI attributes are passed to the *DynamIQ Shared Unit-120* with no modifications, except for translating the following architectural attributes to CHI attributes:

- Allocate hint
- Shareability
- Cacheability


Note

Inner and Outer Cacheability is merged together, as the Cortex-A520 core only allocates memory that is marked as both Inner and Outer cacheable.

Related information

A.1.11 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 190

9.3 Transaction capabilities

The interface between the Cortex-A520 core L2 memory system and the *DynamIQ Shared Unit-120* 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 Cortex-A520 core L2 cache. The table includes values for single-slice L2 cache and dual slice L2 cache configurations.

Table 9-2: Cortex-A520 core transaction capabilities

Attribute	Maximum value	Description
Write issuing capability	40, for single slice 80, for dual slice	Maximum number of outstanding write transactions. **Note:** This value depends on the counting method that is used, but typical values are quoted.
Read issuing capability	31, for single slice 48, for dual slice	Maximum number of outstanding read transactions.
Snoop acceptance capability	29, for single slice 49, for dual slice	Maximum number of outstanding snoops accepted.
DVM issuing capability	7, for single slice 7, for dual slice	Maximum number of outstanding DVM operation transactions.

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

10. Direct access to internal memory

The Cortex-A520 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.

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

Use the **IMPLEMENTATION DEFINED** system registers to select the appropriate memory block and location. The following table shows the System register operations that read the data and the information that the cache data includes.

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

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

10.1 L1 cache encodings

Both the L1 data and instruction caches are 4-way set associative. The size of the configured cache determines the number of sets in each way.

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

To read the data from a particular RAM, write to the appropriate System register using the encoding shown in the table in 10. Direct access to internal memory on page 83.

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

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

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

The cache tag specified is written to IMP_CDBGDR0_EL3.

Related information

A.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0 on page 239
A.4.1 SYS IMP_CDBGL1DCTR, L1 Data Cache Tag Read Operation on page 252
A.4.2 SYS IMP_CDBGL1ICTR, L1 Instruction Cache Tag Read Operation on page 254
A.4.5 SYS IMP_CDBGL1DCDTR, L1 Data Cache Dirty Read Operation on page 258
A.4.6 SYS IMP_CDBGL1DCMR, L1 Data Cache MTE Tag Read Operation on page 259
A.4.9 SYS IMP_CDBGL1DCDR, L1 Data Cache Data Read Operation on page 263
A.4.10 SYS IMP_CDBGL1ICDR, L1 Instruction Cache Data Read Operation on page 264

10.2 L2 cache encodings

The L2 cache is 8-way set associative. The size of the configured cache determines the number of sets in each way.

The encoding that is used to locate the cache data entry for tag and data memory is set in `Xn` in the appropriate `SYS` instruction.

To read the data from a particular RAM, write to the appropriate System register using the encoding shown in the table in 10. Direct access to internal memory on page 83.

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

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

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

The cache tag specified is written to IMP_CDBGDR0_EL3.

Related information

A.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0 on page 239
A.4.4 SYS IMP_CDBGL2CTR, L2 Cache Tag Read Operation on page 256
A.4.8 SYS IMP_CDBGL2CMR, L2 Cache MTE Tag Read Operation on page 262
A.4.12 SYS IMP_CDBGL2CDR, L2 Cache Data Read Operation on page 267

10.3 L2 TLB encodings

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

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

To read the data from a particular TLB, write to the appropriate System register using the encoding shown in the table in 10. Direct access to internal memory on page 83.

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

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

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

The cache tag specified is written to IMP_CDBGDR0_EL3.

Related information

A.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0 on page 239
A.4.3 SYS IMP_CDBGL2TR0, L2 TLB Read Operation 0 on page 255
A.4.7 SYS IMP_CDBGL2TR1, L2 TLB Read Operation 1 on page 260
A.4.11 SYS IMP_CDBGL2TR2, L2 TLB Read Operation 2 on page 266

11. RAS Extension support

The Cortex-A520 core supports the *Reliability, Availability, and Serviceability* (RAS) Extension, including all extensions up to Arm®v9.2-A.

In particular, the Cortex-A520 core supports these RAS Extension features:

- *Fault Handling Interrupts* (FHIs)
- *Error Recovery Interrupts* (ERIs)
- Poison attribute on bus transfers
- Cache protection with *Single Error Detect* (SED) parity
- Cache protection with *Single Error Correct Double Error Detect* (SECDED) *Error Correcting Code* (ECC)
- Error 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. See the *Arm® Architecture Reference Manual for A-profile architecture* for more information on DISR_EL1.

Each of the Cortex-A520 core RAMs has either cache protection with SECDED ECC or cache protection with SED parity, as defined in 11.1 Cache protection behavior on page 86.

Fault detection features are included in groups within the DSU-120 DynamIQ™ cluster and the Cortex-A520 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 Cortex-A520 core and the DSU-120 DynamIQ™ cluster:

- Node 0 includes the shared L3 memory system in the *DynamIQ Shared Unit-120*.
- Node 1 includes the private L1 memory systems in the Cortex-A520 core.
- Node 2 includes the shared L2 memory systems in the complex.

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

For information on the node that includes the shared L3 memory system, see *RAS extension support* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*.

11.1 Cache protection behavior

The configuration of the *Reliability, Availability, and Serviceability* (RAS) Extension that is implemented in the Cortex-A520 core includes cache protection. In this case, the Cortex-A520 core protects against errors that result in a RAM bitcell holding the incorrect value.

The RAMs in the Cortex-A520 core have the following capabilities:

SED parity

Single Error Detect (SED). One bit of parity is applicable to the protected data. The data size is specific for each RAM and depends on the protection granule.

SECDED ECC

Single Error Correct, *Double Error Detect* (SECDED), *Error Correcting Code* (ECC). The data size is specific for each RAM and depends on the protection granule.

The following table indicates which protection type is applied to each RAM in the Cortex-A520 core. The core can progress and remain functionally correct when there is a single-bit error in any RAM.

Table 11-1: RAM cache protection

RAM	ECC or parity
L1 instruction cache data	SED Parity
L1 instruction cache tag	
L1 data cache tag	
L2 *Translation Lookaside Buffer* (TLB)	
L1 data cache data	SECDED ECC
Duplicate L1 data cache tag	
L1 data cache dirty	
L2 cache data	
L2 cache tag	
L2 data buffer	

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

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

- For RAMs with SECDED capability, the core detects, and either reports or defers the error. If the error is in a cache line containing dirty data, then that data might be lost.
- For RAMs with only SED, 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.

11.2 Error containment

The Cortex-A520 core supports error containment for data errors. This means that detected data errors are not silently propagated. Data errors are deferred using data poisoning, ensuring that a consumer is aware of the error. Uncorrectable L1 data cache tag errors and L2 cache tag errors are not containable.

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

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

11.3 Fault detection and reporting

When the Cortex-A520 core detects a fault, it raises a *Fault Handling Interrupt* (FHI) exception or an *Error Recovery Interrupt* (ERI) exception through the fault or the error signals. FHIs and ERIs are reflected in the *Reliability, Availability, and Serviceability* (RAS) registers, which are updated in the node that detects the errors.

Fault handling interrupts

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

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

FHIs from complex *n* are signaled using nCOMPLEXFAULTIRQ[n].

Error recovery interrupts

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

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

ERIs from complex *n* are signaled using nCOMPLEXERRIRQ[n].

11.4 Error detection and reporting

When the Cortex-A520 core consumes an error, it raises different exceptions depending on the error type.

The Cortex-A520 core might raise:

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

11.4.1 Error reporting and performance monitoring

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

The *Performance Monitoring Unit* (PMU) 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. See the *Arm® Architecture Reference Manual for A-profile architecture* for a description of MDCR_EL3.

Related information

18.1 Performance monitors events on page 113

11.5 Error injection

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

Error injection uses the error detection and reporting registers to insert errors. The Cortex-A520 core can inject the following error types:

Corrected errors

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

Deferred errors

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

Uncontainable errors

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

An error can be injected immediately or when a 32-bit counter reaches zero. You can control the value of the counter through the Error Pseudo-fault Generation Countdown Register,

ERR0PFGCDN. The value of the counter decrements on a per clock cycle basis. See the *Arm® Architecture Reference Manual Supplement, Reliability, Availability, and Serviceability (RAS), for A-profile architecture* for more information about ERR0PFGCDN.


Note

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

11.6 AArch64 RAS registers

The 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 11-2: RAS registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ERRIDR_EL1	3	0	C5	C3	0	—	64-bit	Error Record ID Register
ERRSELR_EL1	3	0	C5	C3	1	—	64-bit	Error Record Select Register
ERXFR_EL1	3	0	C5	C4	0	—	64-bit	Selected Error Record Feature Register
ERXCTLR_EL1	3	0	C5	C4	1	—	64-bit	Selected Error Record Control Register
ERXSTATUS_EL1	3	0	C5	C4	2	—	64-bit	Selected Error Record Primary Status Register
ERXADDR_EL1	3	0	C5	C4	3	—	64-bit	Selected Error Record Address Register
ERXPFGF_EL1	3	0	C5	C4	4	—	64-bit	Selected Pseudo-fault Generation Feature register
ERXPFGCTL_EL1	3	0	C5	C4	5	—	64-bit	Selected Pseudo-fault Generation Control register
ERXPFGCDN_EL1	3	0	C5	C4	6	—	64-bit	Selected Pseudo-fault Generation Countdown register
ERXMISC0_EL1	3	0	C5	C5	0	—	64-bit	Selected Error Record Miscellaneous Register 0
ERXMISC1_EL1	3	0	C5	C5	1	—	64-bit	Selected Error Record Miscellaneous Register 1
ERXMISC2_EL1	3	0	C5	C5	2	—	64-bit	Selected Error Record Miscellaneous Register 2
ERXMISC3_EL1	3	0	C5	C5	3	—	64-bit	Selected Error Record Miscellaneous Register 3
DISR_EL1	3	0	C12	C1	1	—	64-bit	Deferred Interrupt Status Register
VSESR_EL2	3	4	C5	C2	3	—	64-bit	Virtual SError Exception Syndrome Register
VDISR_EL2	3	4	C12	C1	1	—	64-bit	Virtual Deferred Interrupt Status Register

11.7 External Complex RAS registers

The summary table provides an overview of all memory-mapped Complex 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 11-3: Complex RAS registers summary

Offset	Name	Reset	Width	Description
0x0	ERR0FR	—	64-bit	Error Record Feature Register
0x8	ERR0CTLR	—	64-bit	Error Record Control Register
0x10	ERR0STATUS	—	64-bit	Error Record Primary Status Register
0x20	ERR0MISC0	—	64-bit	Error Record Miscellaneous Register 0
0x28	ERR0MISC1	—	64-bit	Error Record Miscellaneous Register 1
0x30	ERR0MISC2	—	64-bit	Error Record Miscellaneous Register 2
0x38	ERR0MISC3	—	64-bit	Error Record Miscellaneous Register 3
0x800	ERR0PFGF	—	64-bit	Pseudo-fault Generation Feature Register
0x808	ERR0PFGCTL	—	64-bit	Pseudo-fault Generation Control Register
0x810	ERR0PFGCDN	—	64-bit	Pseudo-fault Generation Countdown Register
0xE00	ERRGSR	—	64-bit	Error Group Status Register
0xE10	ERRIIDR	—	32-bit	Implementation Identification Register
0xFA8	ERRDEVAFF	—	64-bit	Device Affinity Register
0xFBC	ERRDEVARCH	—	32-bit	Device Architecture Register
0xFC8	ERRDEVID	—	32-bit	Device Configuration Register
0xFD0	ERRPIDR4	—	32-bit	Peripheral Identification Register 4
0xFE0	ERRPIDR0	—	32-bit	Peripheral Identification Register 0
0xFE4	ERRPIDR1	—	32-bit	Peripheral Identification Register 1
0xFE8	ERRPIDR2	—	32-bit	Peripheral Identification Register 2
0xFEC	ERRPIDR3	—	32-bit	Peripheral Identification Register 3
0xFF0	ERRCIDR0	—	32-bit	Component Identification Register 0
0xFF4	ERRCIDR1	—	32-bit	Component Identification Register 1
0xFF8	ERRCIDR2	—	32-bit	Component Identification Register 2
0xFFC	ERRCIDR3	—	32-bit	Component Identification Register 3

11.8 External Core RAS registers

The summary table provides an overview of all memory-mapped Core 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 11-4: Core RAS registers summary

Offset	Name	Reset	Width	Description
0x0	ERR0FR	—	64-bit	Error Record Feature Register
0x8	ERR0CTLR	—	64-bit	Error Record Control Register
0x10	ERR0STATUS	—	64-bit	Error Record Primary Status Register
0x20	ERR0MISC0	—	64-bit	Error Record Miscellaneous Register 0
0x28	ERR0MISC1	—	64-bit	Error Record Miscellaneous Register 1
0x30	ERR0MISC2	—	64-bit	Error Record Miscellaneous Register 2
0x38	ERR0MISC3	—	64-bit	Error Record Miscellaneous Register 3
0x800	ERR0PFGF	—	64-bit	Pseudo-fault Generation Feature Register
0x808	ERR0PFGCTL	—	64-bit	Pseudo-fault Generation Control Register
0x810	ERR0PFGCDN	—	64-bit	Pseudo-fault Generation Countdown Register
0xE00	ERRGSR	—	64-bit	Error Group Status Register
0xE10	ERRIIDR	—	32-bit	Implementation Identification Register
0xFA8	ERRDEVAFF	—	64-bit	Device Affinity Register
0xFBC	ERRDEVARCH	—	32-bit	Device Architecture Register
0xFC8	ERRDEVID	—	32-bit	Device Configuration Register
0xFD0	ERRPIDR4	—	32-bit	Peripheral Identification Register 4
0xFE0	ERRPIDR0	—	32-bit	Peripheral Identification Register 0
0xFE4	ERRPIDR1	—	32-bit	Peripheral Identification Register 1
0xFE8	ERRPIDR2	—	32-bit	Peripheral Identification Register 2
0xFEC	ERRPIDR3	—	32-bit	Peripheral Identification Register 3
0xFF0	ERRCIDR0	—	32-bit	Component Identification Register 0
0xFF4	ERRCIDR1	—	32-bit	Component Identification Register 1
0xFF8	ERRCIDR2	—	32-bit	Component Identification Register 2
0xFFC	ERRCIDR3	—	32-bit	Component Identification Register 3

12. Utility bus

The utility bus provides access to control registers for various system components in the *DynamIQ Shared Unit-120* and the cores within the DSU-120 DynamIQ™ cluster. The utility bus is implemented as a 64-bit AMBA AXI5 slave port, and the control registers are memory-mapped onto the utility bus.

The utility bus provides access to the following system functions in the Cortex-A520 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

Information about the *Power Policy Unit* (PPU) registers for the cores in the cluster is provided in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*. For all other registers accessed by the utility bus, see *Utility bus* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*.

12.1 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.


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 in the cluster 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 12-1: 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	Both	B.6 External AMU registers summary on page 679

Base address, n is core instance number	Registers	Security state	Memory map
0x<n>A_0000	Core <n> RAS	Secure	B.3 External Core RAS registers summary on page 518
0x<n>B_0000	Core <n> MPMM	Secure	B.1 External MPMM registers summary on page 460
0x<n>C_0000	Complex RAS, only if core <n> is the first core in the complex	Secure	B.2 External Complex RAS registers summary on page 463
0x<n>D_0000 - 0x<n>F_0000	Reserved	-	-


Note

For more information on utility bus base addresses for system component registers, see the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*.

13. GIC CPU interface

The *Generic Interrupt Controller* (GIC) supports and controls interrupts. The GIC Distributor connects to the Cortex-A520 core through a GIC CPU interface. The GIC CPU interface includes registers to mask, identify, and control the state of interrupts that are forwarded to the core.

Each core in a DSU-120 DynamIQ™ cluster has a GIC CPU interface, which connects to a common external Distributor component.

The GICv4.1 architecture implemented in the Cortex-A520 core supports:

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

The GIC includes interrupt grouping functionality that supports:

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

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

13.1 Disable the GIC CPU interface

The Cortex-A520 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 Cortex-A520 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.

See *Functional integration* in the *Arm®DynamIQ™ Shared Unit-120 Configuration and Integration Manual* for more information on these signals.

13.2 AArch64 GIC system registers

The 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 13-1: GIC system registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICC_PMR_EL1	3	0	C4	C6	0	–	64-bit	Interrupt Controller Interrupt Priority Mask Register
ICV_PMR_EL1	3	0	C4	C6	0	–	64-bit	Interrupt Controller Virtual Interrupt Priority Mask Register
ICC_IAR0_EL1	3	0	C12	C8	0	–	64-bit	Interrupt Controller Interrupt Acknowledge Register 0
ICV_IAR0_EL1	3	0	C12	C8	0	–	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 0
ICC_EOIR0_EL1	3	0	C12	C8	1	–	64-bit	Interrupt Controller End Of Interrupt Register 0
ICV_EOIR0_EL1	3	0	C12	C8	1	–	64-bit	Interrupt Controller Virtual End Of Interrupt Register 0
ICC_HPPIR0_EL1	3	0	C12	C8	2	–	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 0
ICV_HPPIR0_EL1	3	0	C12	C8	2	–	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
ICC_BPR0_EL1	3	0	C12	C8	3	–	64-bit	Interrupt Controller Binary Point Register 0
ICV_BPR0_EL1	3	0	C12	C8	3	–	64-bit	Interrupt Controller Virtual Binary Point Register 0
ICC_AP0R0_EL1	3	0	C12	C8	4	–	64-bit	Interrupt Controller Active Priorities Group 0 Registers
ICV_AP0R0_EL1	3	0	C12	C8	4	–	64-bit	Interrupt Controller Virtual Active Priorities Group 0 Registers
ICC_AP1R0_EL1	3	0	C12	C9	0	–	64-bit	Interrupt Controller Active Priorities Group 1 Registers
ICV_AP1R0_EL1	3	0	C12	C9	0	–	64-bit	Interrupt Controller Virtual Active Priorities Group 1 Registers
ICC_DIR_EL1	3	0	C12	C11	1	–	64-bit	Interrupt Controller Deactivate Interrupt Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICV_DIR_EL1	3	0	C12	C11	1	—	64-bit	Interrupt Controller Deactivate Virtual Interrupt Register
ICC_RPR_EL1	3	0	C12	C11	3	—	64-bit	Interrupt Controller Running Priority Register
ICV_RPR_EL1	3	0	C12	C11	3	—	64-bit	Interrupt Controller Virtual Running Priority Register
ICC_SGI1R_EL1	3	0	C12	C11	5	—	64-bit	Interrupt Controller Software Generated Interrupt Group 1 Register
ICC_ASGI1R_EL1	3	0	C12	C11	6	—	64-bit	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
ICC_SGI0R_EL1	3	0	C12	C11	7	—	64-bit	Interrupt Controller Software Generated Interrupt Group 0 Register
ICC_IAR1_EL1	3	0	C12	C12	0	—	64-bit	Interrupt Controller Interrupt Acknowledge Register 1
ICV_IAR1_EL1	3	0	C12	C12	0	—	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 1
ICC_EOIR1_EL1	3	0	C12	C12	1	—	64-bit	Interrupt Controller End Of Interrupt Register 1
ICV_EOIR1_EL1	3	0	C12	C12	1	—	64-bit	Interrupt Controller Virtual End Of Interrupt Register 1
ICC_HPPIR1_EL1	3	0	C12	C12	2	—	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 1
ICV_HPPIR1_EL1	3	0	C12	C12	2	—	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
ICC_BPR1_EL1	3	0	C12	C12	3	—	64-bit	Interrupt Controller Binary Point Register 1
ICV_BPR1_EL1	3	0	C12	C12	3	—	64-bit	Interrupt Controller Virtual Binary Point Register 1
ICC_CTLR_EL1	3	0	C12	C12	4	—	64-bit	Interrupt Controller Control Register (EL1)
ICV_CTLR_EL1	3	0	C12	C12	4	—	64-bit	Interrupt Controller Virtual Control Register
ICC_SRE_EL1	3	0	C12	C12	5	—	64-bit	Interrupt Controller System Register Enable register (EL1)
ICC_IGRPEN0_EL1	3	0	C12	C12	6	—	64-bit	Interrupt Controller Interrupt Group 0 Enable register
ICV_IGRPEN0_EL1	3	0	C12	C12	6	—	64-bit	Interrupt Controller Virtual Interrupt Group 0 Enable register
ICC_IGRPEN1_EL1	3	0	C12	C12	7	—	64-bit	Interrupt Controller Interrupt Group 1 Enable register
ICV_IGRPEN1_EL1	3	0	C12	C12	7	—	64-bit	Interrupt Controller Virtual Interrupt Group 1 Enable register
ICH_AP0R0_EL2	3	4	C12	C8	0	—	64-bit	Interrupt Controller Hyp Active Priorities Group 0 Registers
ICH_AP1R0_EL2	3	4	C12	C9	0	—	64-bit	Interrupt Controller Hyp Active Priorities Group 1 Registers
ICC_SRE_EL2	3	4	C12	C9	5	—	64-bit	Interrupt Controller System Register Enable register (EL2)
ICH_HCR_EL2	3	4	C12	C11	0	—	64-bit	Interrupt Controller Hyp Control Register
ICH_VTR_EL2	3	4	C12	C11	1	—	64-bit	Interrupt Controller VGIC Type Register
ICH_MISR_EL2	3	4	C12	C11	2	—	64-bit	Interrupt Controller Maintenance Interrupt State Register
ICH_EISR_EL2	3	4	C12	C11	3	—	64-bit	Interrupt Controller End of Interrupt Status Register
ICH_ELRSR_EL2	3	4	C12	C11	5	—	64-bit	Interrupt Controller Empty List Register Status Register
ICH_VMCR_EL2	3	4	C12	C11	7	—	64-bit	Interrupt Controller Virtual Machine Control Register
ICH_LR0_EL2	3	4	C12	C12	0	—	64-bit	Interrupt Controller List Registers
ICH_LR1_EL2	3	4	C12	C12	1	—	64-bit	Interrupt Controller List Registers
ICH_LR2_EL2	3	4	C12	C12	2	—	64-bit	Interrupt Controller List Registers
ICH_LR3_EL2	3	4	C12	C12	3	—	64-bit	Interrupt Controller List Registers
ICC_CTLR_EL3	3	6	C12	C12	4	—	64-bit	Interrupt Controller Control Register (EL3)
ICC_SRE_EL3	3	6	C12	C12	5	—	64-bit	Interrupt Controller System Register Enable register (EL3)
ICC_IGRPEN1_EL3	3	6	C12	C12	7	—	64-bit	Interrupt Controller Interrupt Group 1 Enable register (EL3)

14. Advanced SIMD and floating-point support

The Cortex-A520 core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set without floating-point exception trapping.

The Cortex-A520 core floating-point implementation includes features up to Arm®v9.2-A. BFloat16 floating-point and Int8 matrix multiplication are part of these supported features.

The Cortex-A520 core implements all operations in hardware with support for all combinations of:

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

The Cortex-A520 core supports *Alternate Floating Point* behavior (FEAT_AFP), as part of Arm®v8.7-A and Arm®v9.2-A.

15. Scalable Vector Extensions support

The Cortex-A520 core supports the *Scalable Vector Extension* (SVE) and the *Scalable Vector Extension 2* (SVE2). SVE and SVE2 are intended to complement, not replace, AArch64 Advanced SIMD and floating-point functionality.

SVE is an optional 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 Cortex-A520 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*.

16. 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.

16.1 AArch64 Generic System Control registers

The 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 16-1: Generic System Control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ACTLR_EL1	3	0	C1	C0	1	–	64-bit	Auxiliary Control Register (EL1)
RGSR_EL1	3	0	C1	C0	5	–	64-bit	Random Allocation Tag Seed Register.
GCR_EL1	3	0	C1	C0	6	–	64-bit	Tag Control Register.
TTBR0_EL1	3	0	C2	C0	0	–	64-bit	Translation Table Base Register 0 (EL1)
TTBR1_EL1	3	0	C2	C0	1	–	64-bit	Translation Table Base Register 1 (EL1)
TCR_EL1	3	0	C2	C0	2	–	64-bit	Translation Control Register (EL1)
APIAKeyLo_EL1	3	0	C2	C1	0	–	64-bit	Pointer Authentication Key A for Instruction (bits[63:0])
APIAKeyHi_EL1	3	0	C2	C1	1	–	64-bit	Pointer Authentication Key A for Instruction (bits[127:64])
APIBKeyLo_EL1	3	0	C2	C1	2	–	64-bit	Pointer Authentication Key B for Instruction (bits[63:0])
APIBKeyHi_EL1	3	0	C2	C1	3	–	64-bit	Pointer Authentication Key B for Instruction (bits[127:64])
APDAKeyLo_EL1	3	0	C2	C2	0	–	64-bit	Pointer Authentication Key A for Data (bits[63:0])

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
APDAKeyHi_EL1	3	0	C2	C2	1	–	64-bit	Pointer Authentication Key A for Data (bits[127:64])
APDBKeyLo_EL1	3	0	C2	C2	2	–	64-bit	Pointer Authentication Key B for Data (bits[63:0])
APDBKeyHi_EL1	3	0	C2	C2	3	–	64-bit	Pointer Authentication Key B for Data (bits[127:64])
APGAKeyLo_EL1	3	0	C2	C3	0	–	64-bit	Pointer Authentication Key A for Code (bits[63:0])
APGAKeyHi_EL1	3	0	C2	C3	1	–	64-bit	Pointer Authentication Key A for Code (bits[127:64])
SPSel	3	0	C4	C2	0	–	64-bit	Stack Pointer Select
CurrentEL	3	0	C4	C2	2	–	64-bit	Current Exception Level
PAN	3	0	C4	C2	3	–	64-bit	Privileged Access Never
UAO	3	0	C4	C2	4	–	64-bit	User Access Override
AFSR0_EL1	3	0	C5	C1	0	–	64-bit	Auxiliary Fault Status Register 0 (EL1)
AFSR1_EL1	3	0	C5	C1	1	–	64-bit	Auxiliary Fault Status Register 1 (EL1)
ESR_EL1	3	0	C5	C2	0	–	64-bit	Exception Syndrome Register (EL1)
TFSR_EL1	3	0	C5	C6	0	–	64-bit	Tag Fault Status Register (EL1)
TFSRE0_EL1	3	0	C5	C6	1	–	64-bit	Tag Fault Status Register (EL0).
FAR_EL1	3	0	C6	C0	0	–	64-bit	Fault Address Register (EL1)
PAR_EL1	3	0	C7	C4	0	–	64-bit	Physical Address Register
MAIR_EL1	3	0	C10	C2	0	–	64-bit	Memory Attribute Indirection Register (EL1)
AMAIR_EL1	3	0	C10	C3	0	–	64-bit	Auxiliary Memory Attribute Indirection Register (EL1)
LORSA_EL1	3	0	C10	C4	0	–	64-bit	LORegion Start Address (EL1)
LOREA_EL1	3	0	C10	C4	1	–	64-bit	LORegion End Address (EL1)
LORN_EL1	3	0	C10	C4	2	–	64-bit	LORegion Number (EL1)
LORC_EL1	3	0	C10	C4	3	–	64-bit	LORegion Control (EL1)
LORID_EL1	3	0	C10	C4	7	–	64-bit	LORegionID (EL1)
VBAR_EL1	3	0	C12	C0	0	–	64-bit	Vector Base Address Register (EL1)
ISR_EL1	3	0	C12	C1	0	–	64-bit	Interrupt Status Register
CONTEXTIDR_EL1	3	0	C13	C0	1	–	64-bit	Context ID Register (EL1)
TPIDR_EL1	3	0	C13	C0	4	–	64-bit	EL1 Software Thread ID Register
SCXTNUM_EL1	3	0	C13	C0	7	–	64-bit	EL1 Read/Write Software Context Number
IMP_CPUACTLR_EL1	3	0	C15	C1	0	–	64-bit	CPU Auxiliary Control Register
IMP_CPUACTLR2_EL1	3	0	C15	C1	1	–	64-bit	CPU Auxiliary Control Register 2
IMP_CPUACTLR3_EL1	3	0	C15	C1	2	–	64-bit	CPU Auxiliary Control Register 3
IMP_CMPXACTLR_EL1	3	0	C15	C1	3	–	64-bit	Complex Auxiliary Control Register
IMP_CPUECTLR_EL1	3	0	C15	C1	4	–	64-bit	CPU Extended Control Register
IMP_CMPXECTLR_EL1	3	0	C15	C1	7	–	64-bit	Complex Extended Control Register
IMP_CPUPWRCTLR_EL1	3	0	C15	C2	7	–	64-bit	CPU Power Control Register
IMP_ATCR_EL1	3	0	C15	C7	0	–	64-bit	CPU Auxiliary Translation Control Register
AIDR_EL1	3	1	C0	C0	7	–	64-bit	Auxiliary ID Register
NZCV	3	3	C4	C2	0	–	64-bit	Condition Flags
DAIF	3	3	C4	C2	1	–	64-bit	Interrupt Mask Bits
DIT	3	3	C4	C2	5	–	64-bit	Data Independent Timing

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SSBS	3	3	C4	C2	6	–	64-bit	Speculative Store Bypass Safe
TCO	3	3	C4	C2	7	–	64-bit	Tag Check Override
FPCR	3	3	C4	C4	0	–	64-bit	Floating-point Control Register
FPSR	3	3	C4	C4	1	–	64-bit	Floating-point Status Register
TPIDR_EL0	3	3	C13	C0	2	–	64-bit	EL0 Read/Write Software Thread ID Register
TPIDRRO_EL0	3	3	C13	C0	3	–	64-bit	EL0 Read-Only Software Thread ID Register
SCXTNUM_EL0	3	3	C13	C0	7	–	64-bit	EL0 Read/Write Software Context Number
ACTLR_EL2	3	4	C1	C0	1	–	64-bit	Auxiliary Control Register (EL2)
HACR_EL2	3	4	C1	C1	7	–	64-bit	Hypervisor Auxiliary Control Register
TTBR0_EL2	3	4	C2	C0	0	–	64-bit	Translation Table Base Register 0 (EL2)
TTBR1_EL2	3	4	C2	C0	1	–	64-bit	Translation Table Base Register 1 (EL2)
TCR_EL2	3	4	C2	C0	2	–	64-bit	Translation Control Register (EL2)
VTTBR_EL2	3	4	C2	C1	0	–	64-bit	Virtualization Translation Table Base Register
VTCR_EL2	3	4	C2	C1	2	–	64-bit	Virtualization Translation Control Register
VSTTBR_EL2	3	4	C2	C6	0	–	64-bit	Virtualization Secure Translation Table Base Register
VSTCR_EL2	3	4	C2	C6	2	–	64-bit	Virtualization Secure Translation Control Register
AFSR0_EL2	3	4	C5	C1	0	–	64-bit	Auxiliary Fault Status Register 0 (EL2)
AFSR1_EL2	3	4	C5	C1	1	–	64-bit	Auxiliary Fault Status Register 1 (EL2)
ESR_EL2	3	4	C5	C2	0	–	64-bit	Exception Syndrome Register (EL2)
TFSR_EL2	3	4	C5	C6	0	–	64-bit	Tag Fault Status Register (EL2)
FAR_EL2	3	4	C6	C0	0	–	64-bit	Fault Address Register (EL2)
HPFAR_EL2	3	4	C6	C0	4	–	64-bit	Hypervisor IPA Fault Address Register
MAIR_EL2	3	4	C10	C2	0	–	64-bit	Memory Attribute Indirection Register (EL2)
AMAIR_EL2	3	4	C10	C3	0	–	64-bit	Auxiliary Memory Attribute Indirection Register (EL2)
VBAR_EL2	3	4	C12	C0	0	–	64-bit	Vector Base Address Register (EL2)
CONTEXTIDR_EL2	3	4	C13	C0	1	–	64-bit	Context ID Register (EL2)
TPIDR_EL2	3	4	C13	C0	2	–	64-bit	EL2 Software Thread ID Register
SCXTNUM_EL2	3	4	C13	C0	7	–	64-bit	EL2 Read/Write Software Context Number
IMP_ATCR_EL2	3	4	C15	C7	0	–	64-bit	CPU Auxiliary Translation Control Register
IMP_AVTCR_EL2	3	4	C15	C7	1	–	64-bit	CPU Auxiliary Translation Control Register
ACTLR_EL3	3	6	C1	C0	1	–	64-bit	Auxiliary Control Register (EL3)
SCR_EL3	3	6	C1	C1	0	–	64-bit	Secure Configuration Register
CPTR_EL3	3	6	C1	C1	2	–	64-bit	Architectural Feature Trap Register (EL3)
MDCR_EL3	3	6	C1	C3	1	–	64-bit	Monitor Debug Configuration Register (EL3)
TTBR0_EL3	3	6	C2	C0	0	–	64-bit	Translation Table Base Register 0 (EL3)
TCR_EL3	3	6	C2	C0	2	–	64-bit	Translation Control Register (EL3)
AFSR0_EL3	3	6	C5	C1	0	–	64-bit	Auxiliary Fault Status Register 0 (EL3)
AFSR1_EL3	3	6	C5	C1	1	–	64-bit	Auxiliary Fault Status Register 1 (EL3)
ESR_EL3	3	6	C5	C2	0	–	64-bit	Exception Syndrome Register (EL3)
TFSR_EL3	3	6	C5	C6	0	–	64-bit	Tag Fault Status Register (EL3)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
FAR_EL3	3	6	C6	C0	0	—	64-bit	Fault Address Register (EL3)
MAIR_EL3	3	6	C10	C2	0	—	64-bit	Memory Attribute Indirection Register (EL3)
AMAIR_EL3	3	6	C10	C3	0	—	64-bit	Auxiliary Memory Attribute Indirection Register (EL3)
VBAR_EL3	3	6	C12	C0	0	—	64-bit	Vector Base Address Register (EL3)
RVBAR_EL3	3	6	C12	C0	1	—	64-bit	Reset Vector Base Address Register (if EL3 implemented)
RMR_EL3	3	6	C12	C0	2	—	64-bit	Reset Management Register (EL3)
TPIDR_EL3	3	6	C13	C0	2	—	64-bit	EL3 Software Thread ID Register
SCXTNUM_EL3	3	6	C13	C0	7	—	64-bit	EL3 Read/Write Software Context Number
IMP_ATCR_EL3	3	6	C15	C7	0	—	64-bit	CPU Auxiliary Translation Control Register

17. Debug

The DSU-120 DynamIQ™ cluster provides a debug system that supports both self-hosted and external debug. It has an external DebugBlock component, and integrates various CoreSight debug related components.

The CoreSight debug related components are split into two groups, with some components in the DSU-120 DynamIQ™ cluster, and others in the separate DebugBlock.

The DebugBlock is a dedicated debug component in the DSU-120, separate from the cluster. The DebugBlock operates within a separate power domain, enabling connection to a debugger to be maintained when the cores and the DSU-120 DynamIQ™ cluster are both powered down.

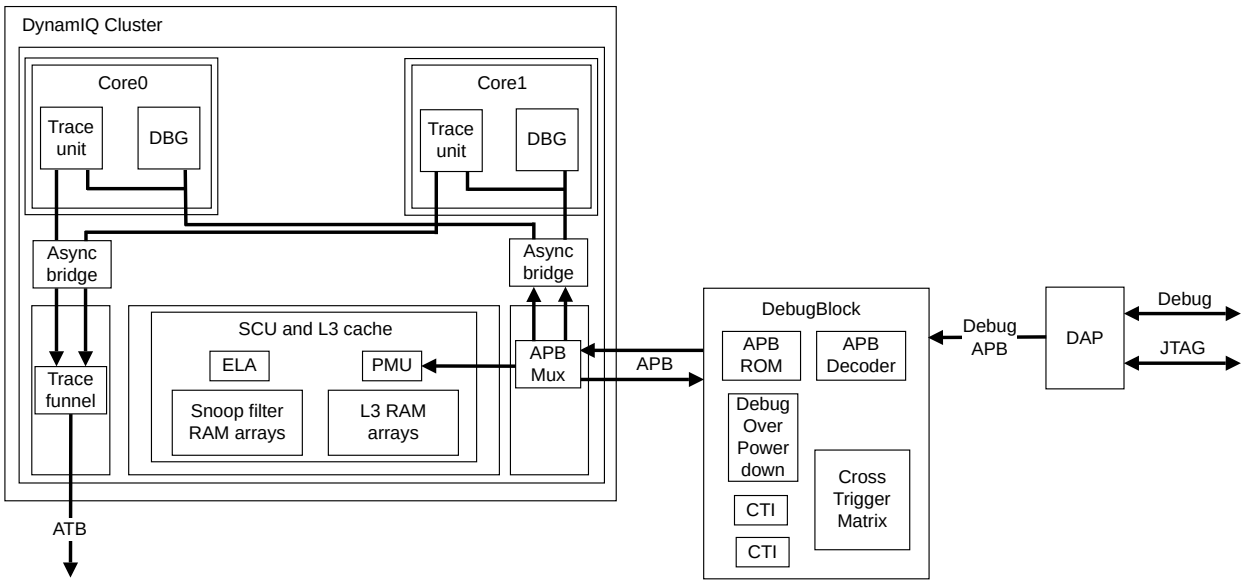
The connection between the cluster and the DebugBlock consists of a pair of *Advanced Peripheral Bus* (APB) interfaces, one in each direction. All debug traffic, except the authentication interface, takes place over this interface as read or write APB transactions. This debug traffic includes register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

The debug system implements the following CoreSight debug components:

- Per-core 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 DSU-120 DynamIQ™ cluster.

Figure 17-1: DSU-120 DynamIQ™ cluster debug components



The primary debug APB interface on the DebugBlock controls the debug components. The APB decoder decodes the requests on this bus before they are sent to the appropriate component in the DebugBlock or in the DSU-120 DynamIQ™ cluster. The per-core CTIs are connected to a CTM.

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

The trace unit in each core outputs trace, which is funneled in the DSU-120 DynamIQ™ cluster down to a single AMBA® 4 ATBv1.1 interface.

See *Debug* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for more information about the DSU-120 DynamIQ™ cluster debug components.

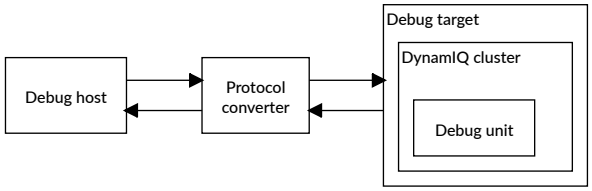
The Cortex-A520 core also supports direct access to internal memory, that is, cache debug. Direct access to internal memory allows software to read the internal memory that the L1 and L2 cache and *Translation Lookaside Buffer* (TLB) structures use. See 10. Direct access to internal memory on page 83 for more information.

17.1 Supported debug methods

The DSU-120 DynamIQ™ cluster along with its associated complexes and cores is part of a debug system that supports both self-hosted and external debug.

The following figure shows a typical external debug system.

Figure 17-2: External debug system



Debug host

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

Protocol converter

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

Debug target

The lowest level of the system implements system support for the protocol converter to access the debug unit. For DSU-120 based devices, the mechanism used to access the debug unit is based on the CoreSight architecture. The DSU-120 DebugBlock is accessed using an APB interface and the debug accesses are then directed to the selected A520 core inside the DSU-120 DynamIQ™ cluster. An example of a debug target is a development system with a test chip or a silicon part with a A520 core.

Debug unit

Helps debugging software that is running on the core:

- DSU-120 and external hardware based around the core.
- Operating systems
- Application software

With the debug unit, you can:

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

For self-hosted debug, the debug target runs debug monitor software that runs on the core in the DSU-120 DynamIQ™ cluster. This way, it does not require expensive interface hardware to connect a second host computer.

17.2 Debug register interfaces

The Cortex-A520 core implements the Arm®v9.2-A Debug architecture. It also supports the Arm®v8.4-A Debug architecture and Arm®v8.3-A Debug over powerdown.

The Debug architecture defines a set of Debug registers. The Debug register interfaces provide access to these registers either from software running on the core or from an external debugger. See *Debug* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for more information.

Related information

5.8 Debug over powerdown on page 61

17.2.1 Core interfaces

All the Debug register groups are both System register based and memory-mapped. System register access allows the Cortex-A520 core to access certain Debug registers directly.

Access to the Debug registers is partitioned as follows:

Debug

You can access the Debug register map using the *Advanced Peripheral Bus* (APB) slave port that connects into the DebugBlock of the *DynamIQ Shared Unit-120*.

Performance monitoring

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

Activity monitoring

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

Trace

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

ELA registers

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

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


Note

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

For information on APB slave port interface, see the *Debug* chapter or the *Interfaces* section in the *Technical overview* chapter of the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*.

Related information

A.3 AArch64 Debug registers summary on page 238
A.7 AArch64 Performance Monitors registers summary on page 357
A.11 AArch64 Activity Monitors registers summary on page 385
A.13 AArch64 Trace unit registers summary on page 410
B.4 External PMU registers summary on page 572
B.5 External Debug registers summary on page 647
B.6 External AMU registers summary on page 679
B.7 External ETE registers summary on page 712

17.2.2 Effects of resets on Debug registers

Cold and Warm resets are generated within the DSU-120 DynamIQ™ cluster and have different effects on the Debug registers.

A Cold reset includes reset of the core logic and the integrated debug functionality. It initializes the core logic, including the trace unit and debug logic.

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

17.2.3 Breakpoints and watchpoints

The Cortex-A520 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.

17.3 Debug events

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

The Cortex-A520 core responds to a debug event in one of the following ways:

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

In the Cortex-A520 core, watchpoint debug events are always synchronous. Memory hint instructions and cache clean operations, except `DC ZVA`, and `DC IVAC` do not generate watchpoint debug events. Store exclusive instructions generate a watchpoint debug event even when the check for the control of exclusive monitor fails. Atomic `CAS` instructions generate a watchpoint debug event even when the compare operation fails.

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

17.4 Debug memory map and debug signals

The debug memory map and debug signals are handled at the DSU-120 DynamIQ™ cluster level.

See *Debug* and *ROM tables* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*.

17.5 ROM table

The Cortex-A520 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 SoC and is for the Cortex-A520 core. There is one ROM table for each complex and ROM tables comply with the *Arm® CoreSight™ Architecture Specification v3.0*.

The *DynamIQ Shared Unit-120* 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 or complex. See *ROM tables* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for more information.

Related information

B.8 External ROM table registers summary on page 772

17.6 CoreSight component identification

Each component associated with the Cortex-A520 core has a unique set of CoreSight™ ID values. The following table shows these values.

Table 17-1: Cortex-A520 CoreSight component identification

Component	Peripheral ID	Component ID	DevType	DevArch	Revision
Debug	0x04200BBD80	0xB105900D	0x15	0x47709A15	r0p2
Trace unit			0x13	0x47715A13	
PMU			0x16	0x47702A16	
ROM table			0x00	0x47700AF7	

17.7 CTI register identification values

The Cortex-A520 core *Cross Trigger Interface* (CTI) registers are located in the DebugBlock of the DSU-120.

For the cluster and core CTI register names and descriptions, see *External CTI registers* in the *Debug* chapter of the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*. Only the core CTI register peripheral ID values will differ from the cluster CTI register peripheral ID values.

The core CTI register peripheral ID values are listed in the following table.

Table 17-2: Core CTI register peripheral ID values

Register	Bitfield position	Bitfield name	Value
CTIPIDR4	[7:4]	SIZE	0b0000

Register	Bitfield position	Bitfield name	Value
	[3:0]	DES_2	0b0100
CTIPIDR3	[7:4]	REVAND	0b0010
	[3:0]	CMOD	0b0000
CTIPIDR2	[7:4]	REVISION	0b0000
	[3]	JEDEC	0b1
	[2:0]	DES_1	0b011
CTIPIDR1	[7:4]	DES_0	0b1011
	[3:0]	PART_1	0b1101
CTIPIDR0	[7:0]	PART_0	0x80

17.8 AArch64 Debug registers

The 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 17-3: Debug registers summary

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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
DBGBVR3_EL1	2	0	C0	C3	4	–	64-bit	Debug Breakpoint Value Registers
DBGBCR3_EL1	2	0	C0	C3	5	–	64-bit	Debug Breakpoint Control Registers
DBGWVR3_EL1	2	0	C0	C3	6	–	64-bit	Debug Watchpoint Value Registers
DBGWCR3_EL1	2	0	C0	C3	7	–	64-bit	Debug Watchpoint Control Registers
DBGBVR4_EL1	2	0	C0	C4	4	–	64-bit	Debug Breakpoint Value Registers
DBGBCR4_EL1	2	0	C0	C4	5	–	64-bit	Debug Breakpoint Control Registers
DBGBVR5_EL1	2	0	C0	C5	4	–	64-bit	Debug Breakpoint Value Registers
DBGBCR5_EL1	2	0	C0	C5	5	–	64-bit	Debug Breakpoint Control Registers
OSECCR_EL1	2	0	C0	C6	2	–	64-bit	OS Lock Exception Catch Control Register
MDRAR_EL1	2	0	C1	C0	0	–	64-bit	Monitor Debug ROM Address Register
OSLAR_EL1	2	0	C1	C0	4	–	64-bit	OS Lock Access Register
OSLSR_EL1	2	0	C1	C1	4	–	64-bit	OS Lock Status Register
OSDLR_EL1	2	0	C1	C3	4	–	64-bit	OS Double Lock Register
DBGPRCR_EL1	2	0	C1	C4	4	–	64-bit	Debug Power Control Register
DBGCLAIMSET_EL1	2	0	C7	C8	6	–	64-bit	Debug CLAIM Tag Set register
DBGCLAIMCLR_EL1	2	0	C7	C9	6	–	64-bit	Debug CLAIM Tag Clear register
DBGAUTHSTATUS_EL1	2	0	C7	C14	6	–	64-bit	Debug Authentication Status register
MDCCSR_EL0	2	3	C0	C1	0	–	64-bit	Monitor DCC Status Register
DBGDTR_EL0	2	3	C0	C4	0	–	64-bit	Debug Data Transfer Register, half-duplex
DBGDTRRX_EL0	2	3	C0	C5	0	–	64-bit	Debug Data Transfer Register, Receive
DBGDTRTX_EL0	2	3	C0	C5	0	–	64-bit	Debug Data Transfer Register, Transmit
TRFCR_EL1	3	0	C1	C2	1	–	64-bit	Trace Filter Control Register (EL1)
MDCR_EL2	3	4	C1	C1	1	–	64-bit	Monitor Debug Configuration Register (EL2)
TRFCR_EL2	3	4	C1	C2	1	–	64-bit	Trace Filter Control Register (EL2)
IMP_CDBGDR0_EL3	3	6	C15	C0	0	–	64-bit	Cache Debug Data Register 0

17.9 External ROM table registers

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

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

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: ROM table registers summary

Offset	Name	Reset	Width	Description
0x0	ROMENTRY0	—	32-bit	Class 0x9 ROM Table Entries
0x4	ROMENTRY1	—	32-bit	Class 0x9 ROM Table Entries
0x8	ROMENTRY2	—	32-bit	Class 0x9 ROM Table Entries
0xC	ROMENTRY3	—	32-bit	Class 0x9 ROM Table Entries
0x10	ROMENTRY4	—	32-bit	Class 0x9 ROM Table Entries
0x14	ROMENTRY5	—	32-bit	Class 0x9 ROM Table Entries
0x18	ROMENTRY6	—	32-bit	Class 0x9 ROM Table Entries
0x1C	ROMENTRY7	—	32-bit	Class 0x9 ROM Table Entries
0xF00	ITCTRL	—	32-bit	Integration Mode Control Register
0xFA0	CLAIMSET	—	32-bit	Claim Tag Set Register
0xFA4	CLAIMCLR	—	32-bit	Claim Tag Clear Register
0xFA8	DEVAFF0	—	32-bit	Device Affinity Register 0
0xFAC	DEVAFF1	—	32-bit	Device Affinity Register 1
0xFB0	LAR	—	32-bit	Software Lock Access Register
0xFB4	LSR	—	32-bit	Software Lock Status Register
0xFB8	AUTHSTATUS	—	32-bit	Authentication Status Register
0xFBC	DEVARCH	—	32-bit	Device Architecture Register
0xFC0	DEVID2	—	32-bit	Device Configuration Register 2
0xFC4	DEVID1	—	32-bit	Device Configuration Register 1
0xFC8	DEVID	—	32-bit	Device Configuration Register
0xFCC	DEVTYPE	—	32-bit	Device Type Register
0xFD0	PIDR4	—	32-bit	Peripheral Identification Register 4
0xFD4	PIDR5	—	32-bit	Peripheral Identification Register 5
0xFD8	PIDR6	—	32-bit	Peripheral Identification Register 6
0xFDC	PIDR7	—	32-bit	Peripheral Identification Register 7
0xFE0	PIDR0	—	32-bit	Peripheral Identification Register 0
0xFE4	PIDR1	—	32-bit	Peripheral Identification Register 1
0xFE8	PIDR2	—	32-bit	Peripheral Identification Register 2
0xFEC	PIDR3	—	32-bit	Peripheral Identification Register 3
0xFF0	CIDR0	—	32-bit	Component Identification Register 0
0xFF4	CIDR1	—	32-bit	Component Identification Register 1
0xFF8	CIDR2	—	32-bit	Component Identification Register 2
0xFFC	CIDR3	—	32-bit	Component Identification Register 3

18. Performance Monitors Extension support

The Cortex-A520 core implements the Performance Monitors Extension, including Arm®v8.7-A performance monitoring features.

The Cortex-A520 core *Performance Monitoring Unit* (PMU):

- Collects events through an event interface from other units in the design. These events are used as triggers for event counters.
- Supports cycle counters through the Performance Monitors Control Register.
- Implements PMU snapshots for context samples.
- Provides six or 20 PMU 64-bit counters that count any of the events available in the core. The absolute counts that are recorded might vary because of pipeline effects. This variation has negligible effect except in cases where the counters are enabled for a very short time.

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

18.1 Performance monitors events

The Cortex-A520 core *Performance Monitoring Unit* (PMU) collects events from other units in the design and uses numbers to reference these events.

18.1.1 Common event PMU events

The following table shows the Cortex-A520 core performance monitors events that are generated and the numbers that the PMU uses to reference the events.

The table also shows the bit position of each event on the event bus. Event numbers that are not listed are reserved.

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


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 18-1: Common event PMU events

Event number	Mnemonic	Description
0x0000	SW_INCR	Instruction architecturally executed, Condition code check pass, software increment The counter counts each write to the AArch64-PMSWINC_EL0 register, for each implemented event counter <n>: If AArch64-PMEVTYPER<n>_EL0.evtCount is 0x0000 then the counter counts each MSR write to AArch64-PMSWINC_EL0 with bit [n] set to 1. If the PE performs two architecturally executed writes to the AArch64-PMSWINC_EL0 register without an intervening Context Synchronization Event, then the counter is incremented twice.
0x0001	L1I_CACHE_REFILL	Level 1 instruction cache refill This event counts any instruction fetch which misses in the cache. The following instructions are not counted: • Cache maintenance instructions • Non-cacheable accesses
0x0002	L1I_TLB_REFILL	Level 1 instruction TLB refill This event counts any refill of the instruction L1 TLB from the L2 TLB, including refills that result in a translation fault. The following instructions are not counted: • TLB maintenance instructions This event counts regardless of whether the *Memory Management Unit* (MMU) is enabled
0x0003	L1D_CACHE_REFILL	Level 1 data cache refill This event counts any load or store operation or translation table walk that causes data to be read from outside the L1 cache, including accesses which do not allocate into the L1 cache. The following instructions are not counted: • Cache maintenance instructions and prefetches • Stores of an entire cache line, even if they make a coherency request outside the L1 cache • Partial cache line writes which do not allocate into the L1 cache • Non-cacheable accesses This event counts the sum of L1D_CACHE_REFILL_RD and L1D_CACHE_REFILL_WR.

Event number	Mnemonic	Description
0x0004	L1D_CACHE	Level 1 data cache access This event counts any load or store operation or translation table walk that looks up in the L1 data cache. In particular, any access that could count the L1D_CACHE_REFILL event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses This event counts the sum of L1D_CACHE_RD and L1D_CACHE_WR.
0x0005	L1D_TLB_REFILL	Level 1 data TLB refill This event counts any refill of the data L1 TLB from the L2 TLB. This includes refills which result in a translation fault. The following instructions are not counted: • TLB maintenance instructions This event counts regardless of whether the MMU is enabled.
0x0006	LD_RETIRED	Instruction architecturally executed, Condition code check pass, load This event counts all load and prefetch instructions, including the Armv8.1-A atomic instructions, other than the ST* variants.
0x0007	ST_RETIRED	Instruction architecturally executed, Condition code check pass, store This event counts all store instructions and the Data Cache Zero by Virtual Address (DC ZVA) instruction. The event includes all the Armv8.1-A atomic instructions. Store-Exclusive instructions that fail are not counted.
0x0008	INST_RETIRED	Instruction architecturally executed This event counts all retired instructions, including those that fail their condition check.
0x0009	EXC_TAKEN	Exception taken The counter counts each exception taken.
0x000A	EXC_RETURN	Instruction architecturally executed, Condition code check pass, exception return The counter counts each architecturally-executed exception return instruction.
0x000B	CID_WRITE_RETIRED	Instruction architecturally executed, Condition code check pass, write to CONTEXTIDR This event only counts writes using the CONTEXTIDR_EL1 mnemonic. Writes to CONTEXTIDR_EL12 and CONTEXTIDR_EL2 are not counted.
0x000C	PC_WRITE_RETIRED	Instruction architecturally executed, Condition code check pass, Software change of the PC This event counts all branches taken and popped from the branch monitor. This excludes exception entries, debug entries, and CCFAIL branches.

Event number	Mnemonic	Description
0x000D	BR_IMMED_RETIRED	Branch instruction architecturally executed, immediate This event counts all branches decoded as immediate branches, taken or not, and popped from the branch monitor. This excludes exception entries, debug entries, and CCFAIL branches.
0x000E	BR_RETURN_RETIRED	Branch instruction architecturally executed, procedure return, taken
0x0010	BR_MIS_PRED	Branch instruction speculatively executed, mispredicted or not predicted This event counts any predictable branch instruction that is mispredicted for either of the following reasons: • Dynamic misprediction • The MMU is off and the branches are statically predicted not taken
0x0011	CPU_CYCLES	Cycle The counter increments on every cycle.
0x0012	BR_PRED	Predictable branch instruction speculatively executed This event counts all predictable branches.
0x0013	MEM_ACCESS	Data memory access This event counts memory accesses due to load or store instructions. Memory accesses are not counted if they are caused by any of the following actions: • Instruction fetches • Cache maintenance instructions • Translation table walks or prefetches This event counts the sum of MEM_ACCESS_RD and MEM_ACCESS_WR.
0x0014	L1I_CACHE	Level 1 instruction cache access This event counts any instruction fetch which accesses the L1 instruction cache. The following instructions are not counted: • Cache maintenance instructions • Non-cacheable accesses
0x0015	L1D_CACHE_WB	Level 1 data cache write-back This event counts any write-back of data from the L1 data cache to L2 cache or L3 cache. The event counts both victim line evictions and snoops, including cache maintenance operations. The following instructions are not counted: • Invalidations which do not result in data being transferred out of the L1 cache • Full-line writes which write to L2 cache without writing L1 cache, such as write-streaming mode

Event number	Mnemonic	Description
0x0016	L2D_CACHE	Level 2 data cache access If the complex is configured with a per-complex L2 cache,this event counts: • Any transaction from the L1 cache which looks up in the L2 cache • Any write-back from the L1 cache to the L2 cache Snoops from outside the core and cache maintenance operations are not counted. If the complex is not configured with a per-complex L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE. If neither a per-complex cache or a cluster cache are configured, then this event is not implemented.
0x0017	L2D_CACHE_REFILL	Level 2 data cache refill If the complex is configured with a per-complex L2 cache, this event counts any cacheable transaction from L1 cache which causes data to be read from outside the core. L2 cache refills that are caused by stashes into L2 cache are counted. If the complex is not configured with a per-complex L2 cache, this event is not implemented.
0x0018	L2D_CACHE_WB	Level 2 data cache write-back If the complex is configured with a per-complex L2 cache, this event counts any write-back of data from the L2 cache to a location outside the complex. The event includes snoops to the L2 cache that return data, regardless of whether they cause an invalidation. Invalidations from the L2 that do not write data outside of the complex and snoops that return data from the L1 cache are not counted. If the core is not configured with a per-complex L2 cache, this event is not implemented.
0x0019	BUS_ACCESS	Bus access This event counts for every beat of data that is transferred over the data channels between the complex and the DynamIQ ®Shared Unit (DSU). If both read and write data beats are transferred on a given cycle, this event is counted twice on that cycle. This event counts the sum of BUS_ACCESS_RD and BUS_ACCESS_WR.
0x001A	MEMORY_ERROR	Local memory error This event counts any correctable or uncorrectable memory error (ECC or parity) in the protected core RAMs.
0x001B	INST_SPEC	Operation speculatively executed This event counts issued instructions, including instructions that are later flushed due to mis-speculation.

Event number	Mnemonic	Description
0x001C	TTBR_WRITE_RETIRED	Instruction architecturally executed, Condition code check pass, write to TTBR This event only counts writes to TTBR0/TTBR1 in AArch32 and TTBR0_EL1/TTBR1_EL1 in AArch64. The following instructions are not counted: • Accesses to TTBR0_EL12/TTBR1_EL12 or TTBR0_EL2/TTBR1_EL2
0x001D	BUS_CYCLES	Bus cycle This event duplicates CPU_CYCLES.
0x001E	CHAIN	CHAIN 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. **Note:** This event is not exported to the trace unit.
0x0020	L2D_CACHE_ALLOCATE	Level 2 data cache allocation without refill If the complex is configured with a per-complex L2 cache, this event counts any full cache line write into the L2 cache that does not cause a linefill. The event includes write-backs from L1 to L2 and full-line writes that do not allocate into the L1 cache. If the complex is not configured with a per-complex L2 cache, this event is not implemented.
0x0021	BR_RETIRED	Instruction architecturally executed, branch Counts all branch instructions, memory-reading and data-processing instructions that explicitly write to the PC, at retirement.
0x0022	BR_MIS_PRED_RETIRED	Branch instruction architecturally executed, mispredicted The counter counts all instructions counted by BR_RETIRED that were not correctly predicted.
0x0023	STALL_FRONTEND	No operation sent for execution due to the frontend The counter counts on any cycle when no operations are issued due to the instruction queue being empty.
0x0024	STALL_BACKEND	No operation sent for execution due to the backend The counter counts on any cycle when no operations are issued due to a pipeline stall.
0x0025	L1D_TLB	Level 1 data TLB access This event counts any load or store operation which accesses the L1 data TLB. If both a load and a store are executed on a cycle, this event counts twice. This event counts regardless of whether the MMU is enabled.

Event number	Mnemonic	Description
0x0026	L1I_TLB	Level 1 instruction TLB access This event counts any instruction fetch which accesses the instruction L1 TLB. This event counts regardless of whether the MMU is enabled.
0x002B	L3D_CACHE	Level 3 data cache access If the complex is configured with a per-complex L2 cache and the cluster is configured with an L3 cache, this event counts for any cacheable read transaction returning data from the DSU, or for any cacheable write to the DSU. If either the complex is configured without a per-complex L2 or the cluster is configured without an L3 cache, this event is not implemented.
0x002D	L2D_TLB_REFILL	Level 2 data TLB refill This event counts on any refill of the L2 TLB, caused by either an instruction or data access. This event does not count if the MMU is disabled.
0x002F	L2D_TLB	Level 2 data TLB access Attributable Level 2 unified TLB access. This event counts on any access to the L2 TLB that is caused by a refill of any of the L1 TLBs. This event does not count if the MMU is disabled.
0x0034	DTLB_WALK	Data TLB access with at least one translation table walk This event counts on any data access which causes L2D_TLB_REFILL to count.
0x0035	ITLB_WALK	Instruction TLB access with at least one translation table walk This event counts on any instruction access which causes L2D_TLB_REFILL to count.
0x0036	LL_CACHE_RD	Last level cache access, read If IMP_CPUECTLR_EL1.EXTLLC is set, this event counts any cacheable read transaction that returns a data source of "interconnect cache". If IMP_CPUECTLR_EL1.EXTLLC is not set, this event is a duplicate of the L*D_CACHE_RD event corresponding to the last level of cache implemented in the cluster. That is: • L3D_CACHE_RD, if both per-complex L2 cache and cluster L3 cache are implemented • L2D_CACHE_RD, if only one of these caches are implemented • L1D_CACHE_RD if neither is implemented.

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

Event number	Mnemonic	Description
0x0040	L1D_CACHE_RD	Level 1 data cache access, read This event counts any load operation or page table walk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_RD event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses **Note:** This event is not exported to the trace unit.
0x0041	L1D_CACHE_WR	Level 1 data cache access, write Counts any store operation which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses **Note:** This event is not exported to the trace unit.
0x0042	L1D_CACHE_REFILL_RD	Level 1 data cache refill, read This event counts any load operation or translation table walk access which causes data to be read from outside the L1 data cache,including accesses which do not allocate into the L1 cache. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x0043	L1D_CACHE_REFILL_WR	Level 1 data cache refill, write This event counts any store operation which causes data to be read from outside the L1 data cache, including accesses which do not allocate into L1 cache. The following instructions are not counted: • Cache maintenance instructions and prefetches. • Stores of an entire cache line, even if they make a coherency request outside the L1 cache • Partial cache line writes which do not allocate into the L1 cache • Non-cacheable accesses **Note:** This event is not exported to the trace unit.
0x0044	L1D_CACHE_REFILL_INNER	Level 1 data cache refill, inner This event counts any L1 data cache linefill, as counted by L1D_CACHE_REFILL, that hits in the L2 cache, L3 cache, or another core in the cluster. **Note:** This event is not exported to the trace unit.
0x0045	L1D_CACHE_REFILL_OUTER	Level 1 data cache refill, outer This event counts any L1 data cache linefill (as counted by L1D_CACHE_REFILL) that does not hit in the L2 cache, L3 cache, or another core in the cluster, and instead obtains data from outside the cluster. **Note:** This event is not exported to the trace unit.
0x0050	L2D_CACHE_RD	Level 2 data cache access, read If the complex is configured with a per-complex L2 cache, this event counts any read transaction from the L1 cache that looks up in the L2 cache. Snoops from outside the complex are not counted. If the complex is configured without a per-complex L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE_RD. If neither a per-complex cache or a cluster cache is configured, this event is not implemented. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x0051	L2D_CACHE_WR	Level 2 data cache access, write If the complex is configured with a per-complex L2 cache, this event counts any write transaction from the L1 cache that looks up in the L2 cache or any write-back from L1 cache that allocates into the L2 cache. Snoops from outside the complex are not counted. If the complex is configured without a per-complex L2 cache, this event is not implemented. **Note:** This event is not exported to the trace unit.
0x0052	L2D_CACHE_REFILL_RD	Level 2 data cache refill, read If the complex is configured with a per-complex L2 cache, this event counts any cacheable read transaction from L1 cache that causes data to be read from outside the complex. L2 cache refills caused by stashes into L2 are not counted. Transactions such as ReadUnique are counted here as read transactions, even though they can be generated by store instructions. If the complex is configured without a per-complex L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE_REFILL_RD. If neither a per-complex cache or a cluster cache is configured, this event is not implemented. **Note:** This event is not exported to the trace unit.
0x0053	L2D_CACHE_REFILL_WR	Level 2 data cache refill, write If the complex is configured with a per-complex L2 cache, this event counts any write transaction from L1 cache that causes data to be read from outside the complex. L2 cache refills caused by stashes into L2 are not counted. Transactions such as ReadUnique are not counted as write transactions. If the complex is configured without a per-core L2 cache, this event is not implemented. **Note:** This event is not exported to the trace unit.
0x0060	BUS_ACCESS_RD	Bus access, read This event counts for every beat of data that is transferred over the read data channel between the complex and the DSU. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x0061	BUS_ACCESS_WR	Bus access, write This event counts for every beat of data that is transferred over the write data channel between the complex and the DSU. **Note:** This event is not exported to the trace unit.
0x0066	MEM_ACCESS_RD	Data memory access, read This event counts memory accesses due to load instructions. The following instructions are not counted: • Instruction fetches • Cache maintenance instructions • Translation table walks • Prefetches **Note:** This event is not exported to the trace unit.
0x0067	MEM_ACCESS_WR	Data memory access, write This event counts memory accesses due to store instructions. The following instructions are not counted: • Instruction fetches • Cache maintenance instructions • Translation table walks • Prefetches **Note:** This event is not exported to the trace unit.
0x006E	STREX_FAIL_SPEC	Exclusive operation speculatively executed, Store-Exclusive fail The counter counts Store-Exclusive instructions speculatively executed that fail to complete a write.
0x006F	STREX_SPEC	Exclusive operation speculatively executed, Store-Exclusive The counter counts Store-Exclusive instructions speculatively executed.
0x0070	LD_SPEC	Operation speculatively executed, load **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x0071	ST_SPEC	Operation speculatively executed, store **Note:** This event is not exported to the trace unit.
0x0072	LDST_SPEC	Operation speculatively executed, load or store **Note:** This event is not exported to the trace unit.
0x0073	DP_SPEC	Operation speculatively executed, integer data processing This event counts retired integer data-processing instructions. **Note:** This event is not exported to the trace unit.
0x0074	ASE_SPEC	Operation speculatively executed, Advanced SIMD This event counts retired Advanced SIMD instructions. **Note:** This event is not exported to the trace unit.
0x0075	VFP_SPEC	Operation speculatively executed, scalar floating-point This event counts retired floating-point instructions. **Note:** This event is not exported to the trace unit.
0x0076	PC_WRITE_SPEC	Operation speculatively executed, Software change of the PC This event counts retired branch instructions. **Note:** This event is not exported to the trace unit.
0x0077	CRYPTO_SPEC	Operation speculatively executed, Cryptographic instruction This event counts retired Cryptographic instructions. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x0078	BR_IMMED_SPEC	Branch speculatively executed, immediate branch This event duplicates BR_IMMED_RETIRED. **Note:** This event is not exported to the trace unit.
0x0079	BR_RETURN_SPEC	Branch speculatively executed, procedure return This event duplicates BR_RETURN_RETIRED. **Note:** This event is not exported to the trace unit.
0x007A	BR_INDIRECT_SPEC	Branch speculatively executed, indirect branch The counter counts indirect branch instructions speculatively executed. This includes software change of the PC other than exception-generating instructions and immediate branch instructions. **Note:** This event is not exported to the trace unit.
0x0086	EXC_IRQ	Exception taken, IRQ **Note:** This event is not exported to the trace unit.
0x0087	EXC_FIQ	Exception taken, FIQ **Note:** This event is not exported to the trace unit.
0x00A0	L3D_CACHE_RD	Level 3 data cache access, read This event counts for any cacheable read transaction returning data from the DSU. If either the complex is configured without a per-complex L2 cache or the cluster is configured without an L3 cache, this event is not implemented. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x00A2	L3D_CACHE_REFILL_RD	Level 3 data cache refill, read If either the complex is configured without a per-complex L2 cache or the cluster is configured without an L3 cache, this event is not implemented. **Note:** This event is not exported to the trace unit.
0x4005	STALL_BACKEND_MEM	Memory stall cycles The counter counts each cycle counted by STALL_BACKEND_MEMBOUND where there is a demand data miss in the last level of cache within the PE clock domain or a non-cacheable data access in progress. If the complex is configured with a per-complex L2 cache, this event is based on L2 cache misses. If the complex is not configured with a per-complex L2 cache, this event is based on L1 data cache misses.
0x4006	L1I_CACHE_LMISS	Level 1 instruction cache long-latency miss The counter counts each access counted by L1I_CACHE that incurs additional latency because it returns instructions from outside the L1 instruction cache.
0x4009	L2D_CACHE_LMISS_RD	Level 2 data cache long-latency read miss If the complex is not configured with a per-complex L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE_LMISS_RD. If neither a per-complex cache or a cluster cache are configured, then this event is not implemented.
0x400B	L3D_CACHE_LMISS_RD	Level 3 data cache long-latency read miss If either the complex is configured without a per-complex L2 or the cluster is configured without an L3 cache, this event is not implemented.
0x400C	TRB_WRAP	Trace buffer current write pointer wrapped The event is generated each time the current write pointer is wrapped to the base pointer.
0x400D	PMU_OVFS	PMU overflow, counters accessible to EL1 and EL0 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 The event is generated when a Trace Buffer Extension Trigger Event occurs.

Event number	Mnemonic	Description
0x400F	PMU_HOVFS	PMU overflow, counters reserved for use by EL2 The event is generated each time an event causes a PMEVCTNR<n>_EL1 counter overflow when PMINTENSET_EL1[n] is set to 1, for each implemented PMU counter n in the range UInt(MDCR_EL2.HPMN) <= n <UInt(PMCR_EL0.N). The 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 The 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 The event is generated each time an event is signaled by ETE external event 1. **Note:** This event is not exported to the trace unit.
0x4012	TRCEXTOUT2	Trace unit external output 2 The 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 The 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 The event is generated each time an event is signaled on CTI output trigger 4.
0x4019	CTI_TRIGOUT5	Cross-trigger Interface output trigger 5 The event is generated each time an event is signaled on CTI output trigger 5.
0x401A	CTI_TRIGOUT6	Cross-trigger Interface output trigger 6 The event is generated each time an event is signaled on CTI output trigger 6.

Event number	Mnemonic	Description
0x401B	CTI_TRIGOUT7	Cross-trigger Interface output trigger 7 The event is generated each time an event is signaled on CTI output trigger 7.
0x4020	LDST_ALIGN_LAT	Access with additional latency from alignment The counter counts each access counted by MEM_ACCESS that, due to the alignment of the address and size of data being accessed, incurred additional latency.
0x4021	LD_ALIGN_LAT	Load with additional latency from alignment The counter counts each Memory-read operation counted by LDST_ALIGN_LAT.
0x4022	ST_ALIGN_LAT	Store with additional latency from alignment The counter counts each Memory-write operation counted by LDST_ALIGN_LAT.
0x4024	MEM_ACCESS_CHECKED	Checked data memory access The counter counts each memory access counted by MEM_ACCESS that is checked by the *Memory Tagging Extension* (MTE).
0x4025	MEM_ACCESS_RD_CHECKED	Checked data memory access, read The counter counts each Memory-read operation counted by MEM_ACCESS_CHECKED.
0x4026	MEM_ACCESS_WR_CHECKED	Checked data memory access, write The counter counts each Memory-write operation counted by MEM_ACCESS_CHECKED.
0x8002	SVE_INST_RETIRED	Instruction architecturally executed, SVE The counter counts architecturally executed SVE instructions.
0x8006	SVE_INST_SPEC	Operation speculatively executed, SVE, including load and store The counter counts speculatively executed operations due to SVE instructions.
0x8014	FP_HP_SPEC	Floating-point operation speculatively executed, half precision
0x8018	FP_SP_SPEC	Floating-point operation speculatively executed, single precision
0x801C	FP_DP_SPEC	Floating-point operation speculatively executed, double precision
0x80E3	ASE_SVE_INT8_SPEC	Integer operation speculatively executed, Advanced SIMD or SVE 8-bit
0x80E7	ASE_SVE_INT16_SPEC	Integer operation speculatively executed, Advanced SIMD or SVE 16-bit
0x80EB	ASE_SVE_INT32_SPEC	Integer operation speculatively executed, Advanced SIMD or SVE 32-bit
0x80EF	ASE_SVE_INT64_SPEC	Integer operation speculatively executed, Advanced SIMD or SVE 64-bit
0x810C	BR_INDNR_TAKEN_RETIRED	Branch instruction architecturally executed, indirect excluding procedure return, taken
0x8110	BR_IMMED_PRED_RETIRED	Branch instruction architecturally executed, predicted immediate
0x8111	BR_IMMED_MIS_PRED_RETIRED	Branch instruction architecturally executed, mispredicted immediate
0x8114	BR_RETURN_PRED_RETIRED	Branch instruction architecturally executed, predicted procedure return
0x8115	BR_RETURN_MIS_PRED_RETIRED	Branch instruction architecturally executed, mispredicted procedure return
0x8116	BR_INDNR_PRED_RETIRED	Branch instruction architecturally executed, predicted indirect excluding procedure return
0x8117	BR_INDNR_MIS_PRED_RETIRED	Branch instruction architecturally executed, mispredicted indirect excluding procedure return
0x811C	BR_PRED_RETIRED	Branch instruction architecturally executed, predicted branch

Event number	Mnemonic	Description
0x811D	BR_IND_RETIRED	Instruction architecturally executed, indirect branch
0x8120	INST_FETCH_PERCYC	Event in progress, INST_FETCH
0x8121	MEM_ACCESS_RD_PERCYC	Event in progress, MEM_ACCESS_RD
0x8124	INST_FETCH	Instruction memory access
0x8125	BUS_REQ_RD_PERCYC	Bus read transactions in progress
0x8128	DTLB_WALK_PERCYC	Event in progress, DTLB_WALK
0x8129	ITLB_WALK_PERCYC	Event in progress, ITLB_WALK
0x8134	DTLB_HWUPD	Data TLB hardware update of translation table
0x8135	ITLB_HWUPD	Instruction TLB hardware update of translation table
0x8136	DTLB_STEP	Data TLB translation table walk, step
0x8137	ITLB_STEP	Instruction TLB translation table walk, step
0x8138	DTLB_WALK_LARGE	Data TLB large page translation table walk Large page is defined as greater than 64KB
0x8139	ITLB_WALK_LARGE	Instruction TLB large page translation table walk Large page is defined as greater than 64KB
0x813A	DTLB_WALK_SMALL	Data TLB small page translation table walk Small page is defined as less than or equal to 64KB
0x813B	ITLB_WALK_SMALL	Instruction TLB small page translation table walk Small page is defined as less than or equal to 64KB
0x813C	DTLB_WALK_RW	Data TLB demand access with at least one translation table walk
0x8154	L1D_CACHE_HWPRF	Level 1 data cache hardware prefetch
0x8155	L2D_CACHE_HWPRF	Level 2 data cache hardware prefetch If the complex is not configured with a per-complex L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE_HWPRF. If neither a per-complex cache or a cluster cache are configured, then this event is not implemented.
0x8156	L3D_CACHE_HWPRF	Level 3 data cache hardware prefetch If either the complex is configured without a per-complex L2 or the cluster is configured without an L3 cache, this event is not implemented.
0x8158	STALL_FRONTEND_MEMBOUND	Frontend stall cycles, memory bound The counter counts each cycle counted by STALL_FRONTEND when no instructions are delivered from the memory system. This includes the cycles counted by STALL_FRONTEND_L1I, STALL_FRONTEND_MEM and STALL_FRONTEND_TLB.

Event number	Mnemonic	Description
0x8159	STALL_FRONTEND_L1I	Frontend stall cycles, level 1 instruction cache The counter counts each cycle counted by STALL_FRONTEND_MEMBOUND when there is a demand instruction miss in the L1 instruction cache. If the complex is configured with a per-complex L2 cache, this event does not count if STALL_FRONTEND_MEM counts. If the complex is not configured with a per-complex L2 cache, this event is not implemented.
0x815B	STALL_FRONTEND_MEM	Frontend stall cycles, last level PE cache or memory The counter counts each cycle counted by STALL_FRONTEND_MEMBOUND when there is a demand instruction miss in the last level of cache within the PE clock domain or a non-cacheable instruction fetch in progress. If the complex is configured with a per-complex L2 cache, this event is based on L2 cache misses. If the complex is not configured with a per-complex L2 cache, this event is based on L1 instruction cache misses.
0x815C	STALL_FRONTEND_TLB	Frontend stall cycles, TLB The counter counts each cycle counted by STALL_FRONTEND_MEMBOUND when there is a demand instruction miss in the L1 instruction TLB
0x8160	STALL_FRONTEND_CPUBOUND	Frontend stall cycles, processor bound The counter counts each cycle counted by STALL_FRONTEND when the frontend is stalled on a frontend processor resource, not including memory. This includes the cycles counted by STALL_FRONTEND_FLOW and STALL_FRONTEND_FLUSH.
0x8161	STALL_FRONTEND_FLOW	Frontend stall cycles, flow control The counter counts each cycle counted by STALL_FRONTEND_CPUBOUND when the frontend is stalled on unavailability of prediction flow resources.
0x8162	STALL_FRONTEND_FLUSH	Frontend stall cycles, flush recovery The counter counts each cycle counted by STALL_FRONTEND_CPUBOUND when the frontend is recovering from a flush
0x8164	STALL_BACKEND_MEMBOUND	Backend stall cycles, memory bound The counter counts each cycle counted by STALL_BACKEND when the backend is waiting for a memory access to complete. This includes the cycles counted by STALL_BACKEND_L1D, STALL_BACKEND_MEM, STALL_BACKEND_ST and STALL_BACKEND_TLB.

Event number	Mnemonic	Description
0x8165	STALL_BACKEND_L1D	Backend stall cycles, level 1 data cache The counter counts each cycle counted by STALL_BACKEND_MEMBOUND when there is a demand data miss in the L1 data cache. If the complex is configured with a per-complex L2 cache, this event does not count if STALL_BACKEND_MEM counts. If the complex is not configured with a per-complex L2 cache, this event is not implemented.
0x8167	STALL_BACKEND_TLB	Backend stall cycles, TLB The counter counts each cycle counted by STALL_BACKEND_MEMBOUND when there is a demand data miss in the L1 data TLB.
0x8168	STALL_BACKEND_ST	Backend stall cycles, store The counter counts each cycle counted by STALL_BACKEND_MEMBOUND when the backend is stalled waiting for a store.
0x816B	STALL_BACKEND_BUSY	Backend stall cycles, backend busy The counter counts each cycle counted by STALL_BACKEND when operations are available from the frontend but the backend is not able to accept an operation because an execution unit is busy.
0x816C	STALL_BACKEND_ILOCK	Backend stall cycles, input dependency The counter counts each cycle counted by STALL_BACKEND when operations are available from the frontend but at least one is not ready to be sent to the backend because of an input dependency.
0x818D	BUS_REQ_RD	Bus request, read
0x81BC	L1D_CACHE_REFILL_HWPRF	Level 1 data cache refill, hardware prefetch
0x81BD	L2D_CACHE_REFILL_HWPRF	Level 2 data cache refill, hardware prefetch

18.1.2 IMPLEMENTATION DEFINED performance monitors events

The Cortex-A520 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.

Table 18-2: Arm IMPLEMENTATION DEFINED PMU events

Event number	Mnemonic	Description
0x00C3	L2D_WS_MODE	L2 cache write streaming mode If the complex is configured with a per-complex L2 cache, this event counts for each cycle where the core is in write streaming mode and is not allocating writes into the L2 cache. If the complex is configured without a per-complex L2 cache, this event counts the cluster cache event, as defined by L3D_WS_MODE. If neither a per-complex cache or a cluster cache is configured, this event is not implemented. **Note:** This event is not exported to the trace unit.
0x00C4	L1D_WS_MODE_ENTRY	L1 data cache entering write streaming mode This event counts for each entry into write streaming mode. **Note:** This event is not exported to the trace unit.
0x00C5	L1D_WS_MODE	L1 data cache write streaming mode This event counts for each cycle where the core is in write streaming mode and is not allocating writes into the L1 data cache. **Note:** This event is not exported to the trace unit.
0x00C7	L3D_WS_MODE	L3 cache write streaming mode This event counts for each cycle where the core is in write streaming mode and is not allocating writes into the L3 cache. If either the complex is configured without a per-complex L2 cache or the cluster is configured without an L3 cache, this event is not implemented. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x00C8	LL_WS_MODE	Last level cache write streaming mode If IMP_CPUECTLR_EL1.EXTLLC is set, this event counts for each cycle where the core is in write streaming mode and is not allocating writes into the system cache. • L3D_WS_MODE, if both per-complex L2 cache and cluster L3 cache are implemented • L2D_WS_MODE, if only one of these caches are implemented • L1D_WS_MODE if neither is implemented. **Note:** This event is not exported to the trace unit.
0x00D0	L2D_WALK_TLB	L2 TLB walk cache access This event does not count if the MMU is disabled. **Note:** This event is not exported to the trace unit.
0x00D1	L2D_WALK_TLB_REFILL	L2 TLB walk cache refill This event does not count if the MMU is disabled. **Note:** This event is not exported to the trace unit.
0x00D4	L2D_S2_TLB	L2 TLB IPA cache access This event counts on each access to the IPA cache. If a single translation table walk needs to make multiple accesses to the IPA cache, each access is counted. If stage 2 translation is disabled, this event does not count. **Note:** This event is not exported to the trace unit.
0x00D5	L2D_S2_TLB_REFILL	L2 TLB IPA cache refill This event counts on each refill of the IPA cache. If a single translation table walk needs to make multiple accesses to the IPA cache, each access that causes a refill is counted. If stage 2 translation is disabled, this event does not count. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x00D6	L2D_CACHE_STASH_DROPPED	L2 cache stash dropped This event counts on each stash request that is received from the interconnect or the Accelerator Coherency Port (ACP), that targets L2 and is dropped due to lack of buffer space to hold the request. If the core is not configured with a per-complex L2 cache, this event is not implemented. **Note:** This event is not exported to the trace unit.
0x00D7	L1D_TLB_REFILL_ETS	L1D TLB refill due to ETS replay This event counts any refill counted by L1D_TLB_REFILL due to ETS replay. **Note:** This event is not exported to the trace unit.
0x00DA	L2D_CACHE_REFILL_HWPRF_SPATIAL	L2 cache refill due to L2 spatial prefetcher This event counts any refill counted by L2D_CACHE_REFILL_HWPRF caused by L2 spatial prefetcher. **Note:** This event is not exported to the trace unit.
0x00DB	L2D_CACHE_REFILL_HWPRF_OFFSET	L2 cache refill due to L2 offset prefetcher This event counts any refill counted by L2D_CACHE_REFILL_HWPRF caused by L2 offset prefetcher. **Note:** This event is not exported to the trace unit.
0x00DC	L2D_CACHE_REFILL_HWPRF_PATTERN	L2 cache refill due to L2 pattern prefetcher This event counts any refill counted by L2D_CACHE_REFILL_HWPRF caused by the L2 pattern prefetcher. **Note:** This event is not exported to the trace unit.
0x00DD	L2D_CACHE_REFILL_HWPRF_TLBD	L2 cache refill due to L2 TLB prefetcher This event counts any refill counted by L2D_CACHE_REFILL_HWPRF caused by the L2 TLB prefetcher. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x00DE	L3D_CACHE_HWPRF_STRIDE	L3 cache access due to L3 stride prefetcher This event counts any access counted by L3D_CACHE_HWPRF caused by the L3 stride prefetcher. **Note:** This event is not exported to the trace unit.
0x00DF	L3D_CACHE_HWPRF_OFFSET	L3 cache access due to L3 offset prefetcher This event counts any access counted by L3D_CACHE_HWPRF caused by the L3 offset prefetcher. **Note:** This event is not exported to the trace unit.
0x00E5	STALL_BACKEND_ILOCK_ADDR	No operation issued due to the backend, input dependency, address This event counts every cycle counted by STALL_BACKEND_ILOCK, where there is an input dependency on an address operand. This type of interlock is caused by a load/store instruction waiting for data to calculate the address. **Note:** This event is not exported to the trace unit.
0x00E6	STALL_BACKEND_ILOCK_VPU	No operation issued due to the backend, input dependency, Vector Processing Unit (VPU). This event counts every cycle counted by STALL_BACKEND_ILOCK, where there is an input dependency on a vector or predicate register. **Note:** This event is not exported to the trace unit.
0x00ED	STALL_BACKEND_BUSY_VPU_HAZARD	No operation issued due to the backend, VPU hazard. This event counts cycles where the core stalls due to contention for the VPU with the other core. **Note:** This event is not exported to the trace unit.
0x00EE	STALL_SLOT_BACKEND_ILOCK	No operation sent for execution on a Slot due to the backend, input dependency For each cycle, this event counts each dispatch slot that does not issue due to an interlock. **Note:** This event is not exported to the trace unit.

Event number	Mnemonic	Description
0x00F0	INST_SPEC_LDST_NUKE	Instruction re-executed, read-after-read hazard This event counts each instruction which re-executes due to read-after-read hazard **Note:** This event is not exported to the trace unit.
0x82FA	DTLB_WALK_HWPRF	Data TLB access, hardware prefetcher This event counts any access counted by DTLB_WALK that is due to a hardware prefetch

18.2 Performance monitors interrupts

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

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

See *Performance Monitors Extension support* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for more information.

18.3 External register access permissions

The Cortex-A520 core supports access to the *Performance Monitoring Unit* (PMU) registers from the system register interface and a memory-mapped interface.

Access to a register depends on:

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

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

18.4 AArch64 Performance Monitors registers

The summary table provides an overview of all Performance Monitors 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 18-3: Performance Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMINTENSET_EL1	3	0	C9	C14	1	—	64-bit	Performance Monitors Interrupt Enable Set register
PMINTENCLR_EL1	3	0	C9	C14	2	—	64-bit	Performance Monitors Interrupt Enable Clear register
PMMIR_EL1	3	0	C9	C14	6	—	64-bit	Performance Monitors Machine Identification Register
PMCR_EL0	3	3	C9	C12	0	—	64-bit	Performance Monitors Control Register
PMCNTENSET_EL0	3	3	C9	C12	1	—	64-bit	Performance Monitors Count Enable Set register
PMCNTENCLR_EL0	3	3	C9	C12	2	—	64-bit	Performance Monitors Count Enable Clear register
PMOVSCLR_EL0	3	3	C9	C12	3	—	64-bit	Performance Monitors Overflow Flag Status Clear Register
PMSWINC_EL0	3	3	C9	C12	4	—	64-bit	Performance Monitors Software Increment register
PMSELR_EL0	3	3	C9	C12	5	—	64-bit	Performance Monitors Event Counter Selection Register
PMCEID0_EL0	3	3	C9	C12	6	—	64-bit	Performance Monitors Common Event Identification register 0
PMCEID1_EL0	3	3	C9	C12	7	—	64-bit	Performance Monitors Common Event Identification register 1
PMCCNTR_EL0	3	3	C9	C13	0	—	64-bit	Performance Monitors Cycle Count Register
PMXEVTYPER_EL0	3	3	C9	C13	1	—	64-bit	Performance Monitors Selected Event Type Register
PMXEVCNTR_EL0	3	3	C9	C13	2	—	64-bit	Performance Monitors Selected Event Count Register
PMUSERENR_EL0	3	3	C9	C14	0	—	64-bit	Performance Monitors User Enable Register
PMOVSSET_EL0	3	3	C9	C14	3	—	64-bit	Performance Monitors Overflow Flag Status Set register
PMEVCNTR0_EL0	3	3	C14	C8	0	—	64-bit	Performance Monitors Event Count Registers
PMEVCNTR1_EL0	3	3	C14	C8	1	—	64-bit	Performance Monitors Event Count Registers
PMEVCNTR2_EL0	3	3	C14	C8	2	—	64-bit	Performance Monitors Event Count Registers
PMEVCNTR3_EL0	3	3	C14	C8	3	—	64-bit	Performance Monitors Event Count Registers
PMEVCNTR4_EL0	3	3	C14	C8	4	—	64-bit	Performance Monitors Event Count Registers
PMEVCNTR5_EL0	3	3	C14	C8	5	—	64-bit	Performance Monitors Event Count Registers
PMEVCNTR6_EL0	3	3	C14	C8	6	—	64-bit	Performance Monitors Event Count Registers
PMEVCNTR7_EL0	3	3	C14	C8	7	—	64-bit	Performance Monitors Event Count Registers
PMEVCNTR8_EL0	3	3	C14	C9	0	—	64-bit	Performance Monitors Event Count Registers
PMEVCNTR9_EL0	3	3	C14	C9	1	—	64-bit	Performance Monitors Event Count Registers
PMEVCNTR10_EL0	3	3	C14	C9	2	—	64-bit	Performance Monitors Event Count Registers
PMEVCNTR11_EL0	3	3	C14	C9	3	—	64-bit	Performance Monitors Event Count Registers

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMEVCNTR12_EL0	3	3	C14	C9	4	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR13_EL0	3	3	C14	C9	5	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR14_EL0	3	3	C14	C9	6	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR15_EL0	3	3	C14	C9	7	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR16_EL0	3	3	C14	C10	0	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR17_EL0	3	3	C14	C10	1	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR18_EL0	3	3	C14	C10	2	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR19_EL0	3	3	C14	C10	3	–	64-bit	Performance Monitors Event Count Registers
PMEVTYPER0_EL0	3	3	C14	C12	0	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER1_EL0	3	3	C14	C12	1	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER2_EL0	3	3	C14	C12	2	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER3_EL0	3	3	C14	C12	3	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER4_EL0	3	3	C14	C12	4	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER5_EL0	3	3	C14	C12	5	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER6_EL0	3	3	C14	C12	6	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER7_EL0	3	3	C14	C12	7	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER8_EL0	3	3	C14	C13	0	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER9_EL0	3	3	C14	C13	1	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER10_EL0	3	3	C14	C13	2	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER11_EL0	3	3	C14	C13	3	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER12_EL0	3	3	C14	C13	4	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER13_EL0	3	3	C14	C13	5	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER14_EL0	3	3	C14	C13	6	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER15_EL0	3	3	C14	C13	7	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER16_EL0	3	3	C14	C14	0	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER17_EL0	3	3	C14	C14	1	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER18_EL0	3	3	C14	C14	2	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER19_EL0	3	3	C14	C14	3	–	64-bit	Performance Monitors Event Type Registers
PMCCFILTR_EL0	3	3	C14	C15	7	–	64-bit	Performance Monitors Cycle Count Filter Register

18.5 External PMU registers

The 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 18-4: PMU registers summary

Offset	Name	Reset	Width	Description
0x0	PMEVCNTR0_EL0	—	64-bit	Performance Monitors Event Count Registers
0x8	PMEVCNTR1_EL0	—	64-bit	Performance Monitors Event Count Registers
0x10	PMEVCNTR2_EL0	—	64-bit	Performance Monitors Event Count Registers
0x18	PMEVCNTR3_EL0	—	64-bit	Performance Monitors Event Count Registers
0x20	PMEVCNTR4_EL0	—	64-bit	Performance Monitors Event Count Registers
0x28	PMEVCNTR5_EL0	—	64-bit	Performance Monitors Event Count Registers
0x30	PMEVCNTR6_EL0	—	64-bit	Performance Monitors Event Count Registers
0x38	PMEVCNTR7_EL0	—	64-bit	Performance Monitors Event Count Registers
0x40	PMEVCNTR8_EL0	—	64-bit	Performance Monitors Event Count Registers
0x48	PMEVCNTR9_EL0	—	64-bit	Performance Monitors Event Count Registers
0x50	PMEVCNTR10_EL0	—	64-bit	Performance Monitors Event Count Registers
0x58	PMEVCNTR11_EL0	—	64-bit	Performance Monitors Event Count Registers
0x60	PMEVCNTR12_EL0	—	64-bit	Performance Monitors Event Count Registers
0x68	PMEVCNTR13_EL0	—	64-bit	Performance Monitors Event Count Registers
0x70	PMEVCNTR14_EL0	—	64-bit	Performance Monitors Event Count Registers
0x78	PMEVCNTR15_EL0	—	64-bit	Performance Monitors Event Count Registers
0x80	PMEVCNTR16_EL0	—	64-bit	Performance Monitors Event Count Registers
0x88	PMEVCNTR17_EL0	—	64-bit	Performance Monitors Event Count Registers
0x90	PMEVCNTR18_EL0	—	64-bit	Performance Monitors Event Count Registers
0x98	PMEVCNTR19_EL0	—	64-bit	Performance Monitors Event Count Registers
0x0F8	PMCCNTR_EL0[31:0]	—	32-bit	Performance Monitors Cycle Counter
0x0FC	PMCCNTR_EL0[63:32]	—	32-bit	Performance Monitors Cycle Counter
0x200	PMPCSR[31:0]	—	32-bit	Program Counter Sample Register
0x204	PMPCSR[63:32]	—	32-bit	Program Counter Sample Register
0x220	PMPCSR[31:0]	—	32-bit	Program Counter Sample Register
0x224	PMPCSR[63:32]	—	32-bit	Program Counter Sample Register
0x208	PMCID1SR	—	32-bit	CONTEXTIDR_EL1 Sample Register
0x228	PMCID1SR	—	32-bit	CONTEXTIDR_EL1 Sample Register
0x20C	PMVIDSR	—	32-bit	VMID Sample Register
0x22C	PMCID2SR	—	32-bit	CONTEXTIDR_EL2 Sample Register
0x400	PMEVTYPER0_EL0	—	32-bit	Performance Monitors Event Type Registers
0x404	PMEVTYPER1_EL0	—	32-bit	Performance Monitors Event Type Registers
0x408	PMEVTYPER2_EL0	—	32-bit	Performance Monitors Event Type Registers
0x40C	PMEVTYPER3_EL0	—	32-bit	Performance Monitors Event Type Registers
0x410	PMEVTYPER4_EL0	—	32-bit	Performance Monitors Event Type Registers
0x414	PMEVTYPER5_EL0	—	32-bit	Performance Monitors Event Type Registers
0x418	PMEVTYPER6_EL0	—	32-bit	Performance Monitors Event Type Registers

Offset	Name	Reset	Width	Description
0x41C	PMEVTYPER7_EL0	—	32-bit	Performance Monitors Event Type Registers
0x420	PMEVTYPER8_EL0	—	32-bit	Performance Monitors Event Type Registers
0x424	PMEVTYPER9_EL0	—	32-bit	Performance Monitors Event Type Registers
0x428	PMEVTYPER10_EL0	—	32-bit	Performance Monitors Event Type Registers
0x42C	PMEVTYPER11_EL0	—	32-bit	Performance Monitors Event Type Registers
0x430	PMEVTYPER12_EL0	—	32-bit	Performance Monitors Event Type Registers
0x434	PMEVTYPER13_EL0	—	32-bit	Performance Monitors Event Type Registers
0x438	PMEVTYPER14_EL0	—	32-bit	Performance Monitors Event Type Registers
0x43C	PMEVTYPER15_EL0	—	32-bit	Performance Monitors Event Type Registers
0x440	PMEVTYPER16_EL0	—	32-bit	Performance Monitors Event Type Registers
0x444	PMEVTYPER17_EL0	—	32-bit	Performance Monitors Event Type Registers
0x448	PMEVTYPER18_EL0	—	32-bit	Performance Monitors Event Type Registers
0x44C	PMEVTYPER19_EL0	—	32-bit	Performance Monitors Event Type Registers
0x47C	PMCCFILTR_EL0	—	32-bit	Performance Monitors Cycle Counter Filter Register
0x600	PMPCSSR	—	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	—	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x610	PMSSSR	—	32-bit	PMU Snapshot Status Register
0x614	PMOVSSR	—	32-bit	PMU Overflow Status Snapshot Register
0x618	PMCCNTSR	—	64-bit	PMU Cycle Counter Snapshot Register
0x620	PMEVCNTSR0	—	64-bit	PMU Event Counter Snapshot Register
0x628	PMEVCNTSR1	—	64-bit	PMU Event Counter Snapshot Register
0x630	PMEVCNTSR2	—	64-bit	PMU Event Counter Snapshot Register
0x638	PMEVCNTSR3	—	64-bit	PMU Event Counter Snapshot Register
0x640	PMEVCNTSR4	—	64-bit	PMU Event Counter Snapshot Register
0x648	PMEVCNTSR5	—	64-bit	PMU Event Counter Snapshot Register
0x650	PMEVCNTSR6	—	64-bit	PMU Event Counter Snapshot Register
0x658	PMEVCNTSR7	—	64-bit	PMU Event Counter Snapshot Register
0x660	PMEVCNTSR8	—	64-bit	PMU Event Counter Snapshot Register
0x668	PMEVCNTSR9	—	64-bit	PMU Event Counter Snapshot Register
0x670	PMEVCNTSR10	—	64-bit	PMU Event Counter Snapshot Register
0x678	PMEVCNTSR11	—	64-bit	PMU Event Counter Snapshot Register
0x680	PMEVCNTSR12	—	64-bit	PMU Event Counter Snapshot Register
0x688	PMEVCNTSR13	—	64-bit	PMU Event Counter Snapshot Register
0x690	PMEVCNTSR14	—	64-bit	PMU Event Counter Snapshot Register
0x698	PMEVCNTSR15	—	64-bit	PMU Event Counter Snapshot Register
0x6A0	PMEVCNTSR16	—	64-bit	PMU Event Counter Snapshot Register
0x6A8	PMEVCNTSR17	—	64-bit	PMU Event Counter Snapshot Register
0x6B0	PMEVCNTSR18	—	64-bit	PMU Event Counter Snapshot Register
0x6B8	PMEVCNTSR19	—	64-bit	PMU Event Counter Snapshot Register
0xC00	PMCNTENSET_EL0	—	32-bit	Performance Monitors Count Enable Set register

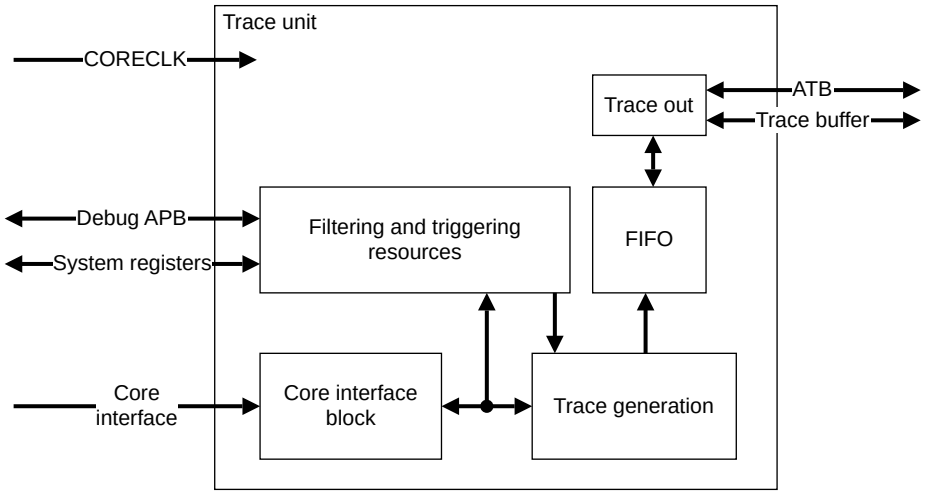
Offset	Name	Reset	Width	Description
0xC20	PMCNTENCLR_EL0	—	32-bit	Performance Monitors Count Enable Clear register
0xC40	PMINTENSET_EL1	—	32-bit	Performance Monitors Interrupt Enable Set register
0xC60	PMINTENCLR_EL1	—	32-bit	Performance Monitors Interrupt Enable Clear register
0xC80	PMOVSCLR_EL0	—	32-bit	Performance Monitors Overflow Flag Status Clear register
0xCC0	PMOVSSET_EL0	—	32-bit	Performance Monitors Overflow Flag Status Set register
0xE00	PMCFGR	—	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_EL0	—	32-bit	Performance Monitors Control Register
0xE20	PMCEID0	—	32-bit	Performance Monitors Common Event Identification register 0
0xE24	PMCEID1	—	32-bit	Performance Monitors Common Event Identification register 1
0xE28	PMCEID2	—	32-bit	Performance Monitors Common Event Identification register 2
0xE2C	PMCEID3	—	32-bit	Performance Monitors Common Event Identification register 3
0xE30	PMSSCR	—	32-bit	PMU Snapshot Capture Register
0xE40	PMMIR	—	32-bit	Performance Monitors Machine Identification Register
0xFA8	PMDEVAFF0	—	32-bit	Performance Monitors Device Affinity register 0
0xFAC	PMDEVAFF1	—	32-bit	Performance Monitors Device Affinity register 1
0xFB0	PMLAR	—	32-bit	Performance Monitors Lock Access Register
0xFB4	PMLSR	—	32-bit	Performance Monitors Lock Status Register
0xFB8	PMAUTHSTATUS	—	32-bit	Performance Monitors Authentication Status register
0xFBC	PMDEVARCH	—	32-bit	Performance Monitors Device Architecture register
0xFC8	PMDEVID	—	32-bit	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	—	32-bit	Performance Monitors Device Type register
0xFD0	PMPIDR4	—	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	—	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	—	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	—	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	—	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	—	32-bit	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	—	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	—	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	—	32-bit	Performance Monitors Component Identification Register 3

19. Embedded Trace Extension support

The Cortex-A520 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 19-1: Trace unit components



Core interface

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

Trace generation

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

Filtering and triggering resources

You can limit the amount of trace data that the trace unit generates by filtering. For example, you can limit trace generation to a certain address range. The trace unit supports other 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 ATB interface or to the trace buffer.

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

19.1 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 A520 core implements.

Table 19-1: Trace unit resources implemented

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

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

19.2 Trace unit generation options

The Cortex-A520 core trace unit implements a set of generation options.

The following table shows the trace generation options that are implemented in the Cortex-A520 core trace unit.

Table 19-2: Trace unit generation options implemented

Description	Configuration
Instruction address size in bytes	8

Description	Configuration
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 because P0 elements	Not implemented
Support for cycle counting in the instruction trace	Implemented
Support for branch broadcast tracing	Implemented
Number of events that are supported in the trace	4
Return stack support	Implemented
Tracing of SError exception support	Implemented
Instruction trace cycle counting minimum threshold	4
Size of Trace ID	7 bits
Synchronization period support	Read/write
Global timestamp size	64 bits
Number of cores available for tracing	1
ATB trigger support	Implemented
Low-power behavior override	Implemented
Stall control support	Not implemented
Support for overflow avoidance	Not implemented
Support for using CONTEXTIDR_EL2 in *Virtual Machine IDentifier* (VMID) comparator	Implemented

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

19.3 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.

19.4 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.

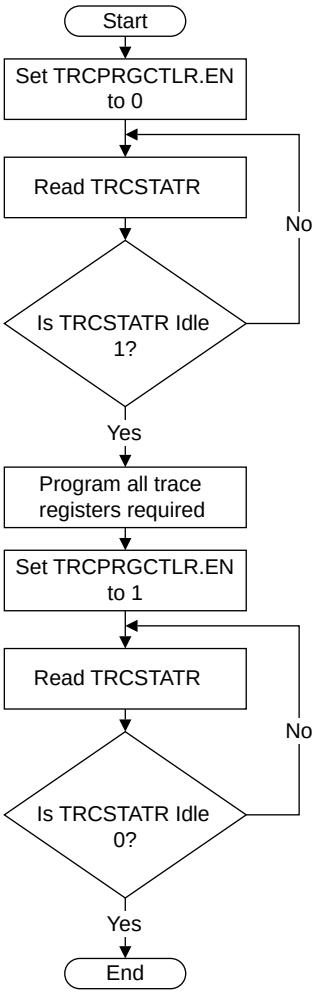
The core does not have to be in debug state when you program the trace unit registers. When you program the trace unit registers, you must enable all the changes at the same time. Otherwise, if you program the counter, it might start to count based on incorrect events before the correct setup is in place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

See the *Arm® Architecture Reference Manual for A-profile architecture* for more information about the following trace unit registers:

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

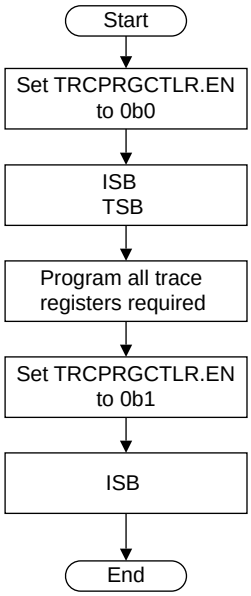
The following figure shows the flow for programming trace unit registers using the Debug APB interface:

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



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

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



19.5 Trace unit register interfaces

The Cortex-A520 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.

19.6 Interaction with the Performance Monitoring Unit and Debug

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

Interaction with the PMU

The Cortex-A520 core includes a PMU that enables events, such as cache misses and executed instructions, to be counted over time.

The PMU and trace unit function together.

Use of PMU events by the trace unit

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

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

Related information

18. Performance Monitors Extension support on page 113
18.1 Performance monitors events on page 113

19.7 Embedded Trace Extension events

The Cortex-A520 core trace unit collects events from other units in the design and uses numbers to reference these events.

The exported events are listed in the table in 18.1.1 Common event PMU events on page 113.

19.8 AArch64 Trace unit registers

The summary table provides an overview of all Trace unit 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 19-3: Trace unit registers summary

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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCSEQEVR2	2	1	C0	C2	4	–	64-bit	Sequencer State Transition Control Register <n>
TRCIDR10	2	1	C0	C2	6	–	64-bit	ID Register 10
TRCSTATR	2	1	C0	C3	0	–	64-bit	Trace Status Register
TRCIDR11	2	1	C0	C3	6	–	64-bit	ID Register 11
TRCCONFIGR	2	1	C0	C4	0	–	64-bit	Trace Configuration Register
TRCCNTCTLR0	2	1	C0	C4	5	–	64-bit	Counter Control Register <n>
TRCIDR12	2	1	C0	C4	6	–	64-bit	ID Register 12
TRCCNTCTLR1	2	1	C0	C5	5	–	64-bit	Counter Control Register <n>
TRCIDR13	2	1	C0	C5	6	–	64-bit	ID Register 13
TRCAUXCTLR	2	1	C0	C6	0	–	64-bit	Auxiliary Control Register
TRCSEQRSTEVR	2	1	C0	C6	4	–	64-bit	Sequencer Reset Control Register
TRCSEQSTR	2	1	C0	C7	4	–	64-bit	Sequencer State Register
TRCEVENTCTL0R	2	1	C0	C8	0	–	64-bit	Event Control 0 Register
TRCEXTINSELR0	2	1	C0	C8	4	–	64-bit	External Input Select Register <n>
TRCCNTVR0	2	1	C0	C8	5	–	64-bit	Counter Value Register <n>
TRCIDR0	2	1	C0	C8	7	–	64-bit	ID Register 0
TRCEVENTCTL1R	2	1	C0	C9	0	–	64-bit	Event Control 1 Register
TRCEXTINSELR1	2	1	C0	C9	4	–	64-bit	External Input Select Register <n>
TRCCNTVR1	2	1	C0	C9	5	–	64-bit	Counter Value Register <n>
TRCIDR1	2	1	C0	C9	7	–	64-bit	ID Register 1
TRCRSR	2	1	C0	C10	0	–	64-bit	Resources Status Register
TRCEXTINSELR2	2	1	C0	C10	4	–	64-bit	External Input Select Register <n>
TRCIDR2	2	1	C0	C10	7	–	64-bit	ID Register 2
TRCSTALLCTLR	2	1	C0	C11	0	–	64-bit	Stall Control Register
TRCEXTINSELR3	2	1	C0	C11	4	–	64-bit	External Input Select Register <n>
TRCIDR3	2	1	C0	C11	7	–	64-bit	ID Register 3
TRCTSCTLR	2	1	C0	C12	0	–	64-bit	Timestamp Control Register
TRCIDR4	2	1	C0	C12	7	–	64-bit	ID Register 4
TRCSYNCPR	2	1	C0	C13	0	–	64-bit	Synchronization Period Register
TRCIDR5	2	1	C0	C13	7	–	64-bit	ID Register 5
TRCCCCTLR	2	1	C0	C14	0	–	64-bit	Cycle Count Control Register
TRCIDR6	2	1	C0	C14	7	–	64-bit	ID Register 6
TRCBBCTLR	2	1	C0	C15	0	–	64-bit	Branch Broadcast Control Register
TRCIDR7	2	1	C0	C15	7	–	64-bit	ID Register 7
TRCSSCCR0	2	1	C1	C0	2	–	64-bit	Single-shot Comparator Control Register <n>
TRCOSLSR	2	1	C1	C1	4	–	64-bit	Trace OS Lock Status Register
TRCRSCTLR2	2	1	C1	C2	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR3	2	1	C1	C3	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR4	2	1	C1	C4	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR5	2	1	C1	C5	0	–	64-bit	Resource Selection Control Register <n>

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCRSCTLR6	2	1	C1	C6	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR7	2	1	C1	C7	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR8	2	1	C1	C8	0	–	64-bit	Resource Selection Control Register <n>
TRCSSCSR0	2	1	C1	C8	2	–	64-bit	Single-shot Comparator Control Status Register <n>
TRCRSCTLR9	2	1	C1	C9	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR10	2	1	C1	C10	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR11	2	1	C1	C11	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR12	2	1	C1	C12	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR13	2	1	C1	C13	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR14	2	1	C1	C14	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR15	2	1	C1	C15	0	–	64-bit	Resource Selection Control Register <n>
TRCACVR0	2	1	C2	C0	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR0	2	1	C2	C0	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR1	2	1	C2	C2	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR1	2	1	C2	C2	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR2	2	1	C2	C4	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR2	2	1	C2	C4	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR3	2	1	C2	C6	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR3	2	1	C2	C6	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR4	2	1	C2	C8	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR4	2	1	C2	C8	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR5	2	1	C2	C10	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR5	2	1	C2	C10	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR6	2	1	C2	C12	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR6	2	1	C2	C12	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR7	2	1	C2	C14	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR7	2	1	C2	C14	2	–	64-bit	Address Comparator Access Type Register <n>
TRCCIDCVR0	2	1	C3	C0	0	–	64-bit	Context Identifier Comparator Value Registers <n>
TRCVMIDCVR0	2	1	C3	C0	1	–	64-bit	Virtual Context Identifier Comparator Value Register <n>
TRCCIDCCTLR0	2	1	C3	C0	2	–	64-bit	Context Identifier Comparator Control Register 0
TRCVMIDCCTLR0	2	1	C3	C2	2	–	64-bit	Virtual Context Identifier Comparator Control Register 0
TRCDEVID	2	1	C7	C2	7	–	64-bit	Device Configuration Register
TRCCLAIMSET	2	1	C7	C8	6	–	64-bit	Claim Tag Set Register
TRCCLAIMCLR	2	1	C7	C9	6	–	64-bit	Claim Tag Clear Register
TRCAUTHSTATUS	2	1	C7	C14	6	–	64-bit	Authentication Status Register
TRCDEVARCH	2	1	C7	C15	6	–	64-bit	Device Architecture Register

19.9 External ETE registers

The 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 19-4: ETE registers summary

Offset	Name	Reset	Width	Description
0x004	TRCPRGCTLR	—	32-bit	Programming Control Register
0x00C	TRCSTATR	—	32-bit	Trace Status Register
0x010	TRCCONFIGR	—	32-bit	Trace Configuration Register
0x018	TRCAUXCTLR	—	32-bit	Auxiliary Control Register
0x020	TRCEVENTCTL0R	—	32-bit	Event Control 0 Register
0x024	TRCEVENTCTL1R	—	32-bit	Event Control 1 Register
0x028	TRCRSR	—	32-bit	Resources Status Register
0x02C	TRCSTALLCTLR	—	32-bit	Stall Control Register
0x030	TRCTSCTLR	—	32-bit	Timestamp Control Register
0x034	TRCSYNCPR	—	32-bit	Synchronization Period Register
0x038	TRCCCCTLR	—	32-bit	Cycle Count Control Register
0x03C	TRCBBCTLR	—	32-bit	Branch Broadcast Control Register
0x040	TRCTRACEIDR	—	32-bit	Trace ID Register
0x044	TRCQCTLR	—	32-bit	Q Element Control Register
0x080	TRCVICTLR	—	32-bit	ViewInst Main Control Register
0x084	TRCVIIECTLR	—	32-bit	ViewInst Include/Exclude Control Register
0x088	TRCVISSCTLR	—	32-bit	ViewInst Start/Stop Control Register
0x100	TRCSEQEVR0	—	32-bit	Sequencer State Transition Control Register <n>
0x104	TRCSEQEVR1	—	32-bit	Sequencer State Transition Control Register <n>
0x108	TRCSEQEVR2	—	32-bit	Sequencer State Transition Control Register <n>
0x118	TRCSEQRSTEVR	—	32-bit	Sequencer Reset Control Register
0x11C	TRCSEQSTR	—	32-bit	Sequencer State Register
0x120	TRCEXTINSELR0	—	32-bit	External Input Select Register <n>
0x124	TRCEXTINSELR1	—	32-bit	External Input Select Register <n>
0x128	TRCEXTINSELR2	—	32-bit	External Input Select Register <n>
0x12C	TRCEXTINSELR3	—	32-bit	External Input Select Register <n>
0x140	TRCCNTRLDVR0	—	32-bit	Counter Reload Value Register <n>
0x144	TRCCNTRLDVR1	—	32-bit	Counter Reload Value Register <n>

Offset	Name	Reset	Width	Description
0x150	TRCCNTCTLR0	—	32-bit	Counter Control Register <n>
0x154	TRCCNTCTLR1	—	32-bit	Counter Control Register <n>
0x160	TRCCNTVR0	—	32-bit	Counter Value Register <n>
0x164	TRCCNTVR1	—	32-bit	Counter Value Register <n>
0x180	TRCIDR8	—	32-bit	ID Register 8
0x184	TRCIDR9	—	32-bit	ID Register 9
0x188	TRCIDR10	—	32-bit	ID Register 10
0x18C	TRCIDR11	—	32-bit	ID Register 11
0x190	TRCIDR12	—	32-bit	ID Register 12
0x194	TRCIDR13	—	32-bit	ID Register 13
0x1C0	TRCIMSPEC0	—	32-bit	IMP DEF Register 0
0x1E0	TRCIDR0	—	32-bit	ID Register 0
0x1E4	TRCIDR1	—	32-bit	ID Register 1
0x1E8	TRCIDR2	—	32-bit	ID Register 2
0x1EC	TRCIDR3	—	32-bit	ID Register 3
0x1F0	TRCIDR4	—	32-bit	ID Register 4
0x1F4	TRCIDR5	—	32-bit	ID Register 5
0x1F8	TRCIDR6	—	32-bit	ID Register 6
0x1FC	TRCIDR7	—	32-bit	ID Register 7
0x208	TRCRSCTLR2	—	32-bit	Resource Selection Control Register <n>
0x20C	TRCRSCTLR3	—	32-bit	Resource Selection Control Register <n>
0x210	TRCRSCTLR4	—	32-bit	Resource Selection Control Register <n>
0x214	TRCRSCTLR5	—	32-bit	Resource Selection Control Register <n>
0x218	TRCRSCTLR6	—	32-bit	Resource Selection Control Register <n>
0x21C	TRCRSCTLR7	—	32-bit	Resource Selection Control Register <n>
0x220	TRCRSCTLR8	—	32-bit	Resource Selection Control Register <n>
0x224	TRCRSCTLR9	—	32-bit	Resource Selection Control Register <n>
0x228	TRCRSCTLR10	—	32-bit	Resource Selection Control Register <n>
0x22C	TRCRSCTLR11	—	32-bit	Resource Selection Control Register <n>
0x230	TRCRSCTLR12	—	32-bit	Resource Selection Control Register <n>
0x234	TRCRSCTLR13	—	32-bit	Resource Selection Control Register <n>
0x238	TRCRSCTLR14	—	32-bit	Resource Selection Control Register <n>
0x23C	TRCRSCTLR15	—	32-bit	Resource Selection Control Register <n>
0x280	TRCSSCCR0	—	32-bit	Single-shot Comparator Control Register <n>
0x2A0	TRCSSCSR0	—	32-bit	Single-shot Comparator Control Status Register <n>
0x304	TRCOSLSR	—	32-bit	Trace OS Lock Status Register
0x310	TRCPDCR	—	32-bit	PowerDown Control Register
0x314	TRCPDSR	—	32-bit	PowerDown Status Register
0x400	TRCACVR0	—	64-bit	Address Comparator Value Register <n>
0x408	TRCACVR1	—	64-bit	Address Comparator Value Register <n>

Offset	Name	Reset	Width	Description
0x410	TRCACVR2	—	64-bit	Address Comparator Value Register <n>
0x418	TRCACVR3	—	64-bit	Address Comparator Value Register <n>
0x420	TRCACVR4	—	64-bit	Address Comparator Value Register <n>
0x428	TRCACVR5	—	64-bit	Address Comparator Value Register <n>
0x430	TRCACVR6	—	64-bit	Address Comparator Value Register <n>
0x438	TRCACVR7	—	64-bit	Address Comparator Value Register <n>
0x480	TRCACATR0	—	64-bit	Address Comparator Access Type Register <n>
0x488	TRCACATR1	—	64-bit	Address Comparator Access Type Register <n>
0x490	TRCACATR2	—	64-bit	Address Comparator Access Type Register <n>
0x498	TRCACATR3	—	64-bit	Address Comparator Access Type Register <n>
0x4A0	TRCACATR4	—	64-bit	Address Comparator Access Type Register <n>
0x4A8	TRCACATR5	—	64-bit	Address Comparator Access Type Register <n>
0x4B0	TRCACATR6	—	64-bit	Address Comparator Access Type Register <n>
0x4B8	TRCACATR7	—	64-bit	Address Comparator Access Type Register <n>
0x600	TRCCIDCVR0	—	64-bit	Context Identifier Comparator Value Registers <n>
0x640	TRCVMIDCVR0	—	64-bit	Virtual Context Identifier Comparator Value Register <n>
0x680	TRCCIDCCTLR0	—	32-bit	Context Identifier Comparator Control Register 0
0x688	TRCVMIDCCTLR0	—	32-bit	Virtual Context Identifier Comparator Control Register 0
0xEE4	TRCITATBIDR	—	32-bit	Trace Intergration ATB Identification Register
0xEEC	TRCITATBDATAR	—	32-bit	Trace Integration Test ATB Data Register 0
0xEF4	TRCITATBINR	—	32-bit	Trace Integration ATB In Register
0xEFC	TRCITATBOUTR	—	32-bit	Trace Integration ATB Out Register
0xF00	TRCITCTRL	—	32-bit	Integration Mode Control Register
0xFA0	TRCCLAIMSET	—	32-bit	Claim Tag Set Register
0xFA4	TRCCLAIMCLR	—	32-bit	Claim Tag Clear Register
0xFA8	TRCDEVAFF	—	64-bit	Device Affinity Register
0xFB0	TRCLAR	—	32-bit	Lock Access Register
0xFB4	TRCLSR	—	32-bit	Lock Status Register
0xFB8	TRCAUTHSTATUS	—	32-bit	Authentication Status Register
0xFBC	TRCDEVARCH	—	32-bit	Device Architecture Register
0xFC0	TRCDEVID2	—	32-bit	Device Configuration Register 2
0xFC4	TRCDEVID1	—	32-bit	Device Configuration Register 1
0xFC8	TRCDEVID	—	32-bit	Device Configuration Register
0xFCC	TRCDEVTYPE	—	32-bit	Device Type Register
0xFD0	TRCPIDR4	—	32-bit	Peripheral Identification Register 4
0xFD4	TRCPIDR5	—	32-bit	Peripheral Identification Register 5
0xFD8	TRCPIDR6	—	32-bit	Peripheral Identification Register 6
0xFDC	TRCPIDR7	—	32-bit	Peripheral Identification Register 7
0xFE0	TRCPIDR0	—	32-bit	Peripheral Identification Register 0
0xFE4	TRCPIDR1	—	32-bit	Peripheral Identification Register 1

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

20. Trace Buffer Extension support

The Cortex-A520 core implements the *TRace Buffer Extension* (TRBE). The TRBE writes the program flow trace generated by the trace unit directly to memory. The TRBE is programmed through System registers.

When enabled, the TRBE can:

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

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

20.1 Program and read the trace buffer registers

You 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.

See the *Arm® Architecture Reference Manual for A-profile architecture* for information on the TRBE register behaviors and access mechanisms.

20.2 Trace buffer register interface

The Cortex-A520 core supports a System register interface to *TRace Buffer Extension* (TRBE) registers.

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

20.3 AArch64 Trace Buffer Extension registers

The summary table provides an overview of all Trace Buffer Extension 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 20-1: Trace Buffer Extension registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRBLIMITR_EL1	3	0	C9	C11	0	–	64-bit	Trace Buffer Limit Address Register
TRBPTR_EL1	3	0	C9	C11	1	–	64-bit	Trace Buffer Write Pointer Register
TRBBASER_EL1	3	0	C9	C11	2	–	64-bit	Trace Buffer Base Address Register
TRBSR_EL1	3	0	C9	C11	3	–	64-bit	Trace Buffer Status/syndrome Register
TRBMAR_EL1	3	0	C9	C11	4	–	64-bit	Trace Buffer Memory Attribute Register
TRBTRG_EL1	3	0	C9	C11	6	–	64-bit	Trace Buffer Trigger Counter Register
TRBIDR_EL1	3	0	C9	C11	7	–	64-bit	Trace Buffer ID Register

21. Activity Monitors Extension support

The Cortex-A520 core implements the Activity Monitors Extension to the Arm®v8.4-A architecture. Activity monitoring has features similar to performance monitoring features, but is intended for system management use whereas performance monitoring is aimed at user and debug applications.

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

The Cortex-A520 core implements seven counters in two groups, each of which is a 64-bit counter that counts a fixed event. Group 0 has four counters 0-3, and Group 1 has three counters 0-2.

21.1 Activity monitors access

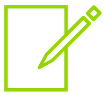
The Cortex-A520 core supports access to activity monitors from the System register interface and supports read-only memory-mapped access using the utility bus interface.

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



AMUSERENR_EL0.EN is configurable at EL1, EL2, and EL3. All other controls, as well as the value of the counters, are configurable only at the highest implemented Exception level.

For a detailed description of access controls for the registers, see the *Arm® Architecture Reference Manual for A-profile architecture*.

System register access

The activity monitors are accessible using the `MRS` and `MSR` instructions.

External memory-mapped access

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

The base address for *Activity Monitoring Unit* (AMU) registers on the utility bus interface is `0x<n>90000`, where *n* is the Cortex-A520 core instance number in the DSU-120 DynamIQ™ cluster.

These registers are treated as RAZ/WI if either:

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

21.2 Activity monitors counters

The Cortex-A520 core implements seven activity monitors counters 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.
- The activity monitor counters are reset to zero on a Cold reset of the power domain of the core. When the core is not in reset, activity monitoring is available.

21.3 Activity monitors events

Activity monitors events in the Cortex-A520 core are fixed, and they map to the activity monitors counters.

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

Table 21-1: Mapping of counters to fixed events

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR00	CPU_CYCLES	0x0011	Core frequency cycles
AMEVCNTR01	CNT_CYCLES	0x4004	Constant frequency cycles
AMEVCNTR02	INST_RETIRED	0x0008	Instruction architecturally executed Increments for every instruction that is executed architecturally, including instructions that fail their condition code check
AMEVCNTR03	STALL_BACKEND_MEM	0x4005	Memory stall cycles Increments for each cycle in which the core is unable to dispatch instructions from the front end to the back end due to a back end stall caused by a miss in the last level of cache within the core clock domain

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR10	GEAR0_MPMM_ATHR_EXCEEDED	0x0300	*Maximum Power Mitigation System* (MPMM) Gear 0 Increments for each period where core activity is above the throttling threshold for gear 0 Reserved
AMEVCNTR11	GEAR1_MPMM_ATHR_EXCEEDED	0x0301	MPMM Gear 1 Increments for each period where core activity is above the throttling threshold for gear 1 Reserved
AMEVCNTR12	GEAR2_MPMM_ATHR_EXCEEDED	0x0302	MPMM Gear 2 Increments for each period where core activity is above the throttling threshold for gear 2 Reserved

Related information

5.6.1 Maximum Power Mitigation Mechanism on page 58

21.4 AArch64 Activity Monitors registers

The summary table provides an overview of all Activity Monitors 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 21-2: Activity Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMCR_EL0	3	3	C13	C2	0	—	64-bit	Activity Monitors Control Register
AMCFGR_EL0	3	3	C13	C2	1	—	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	3	3	C13	C2	2	—	64-bit	Activity Monitors Counter Group Configuration Register
AMUSERENR_EL0	3	3	C13	C2	3	—	64-bit	Activity Monitors User Enable Register
AMCNTENCLR0_EL0	3	3	C13	C2	4	—	64-bit	Activity Monitors Count Enable Clear Register 0
AMCNTENSET0_EL0	3	3	C13	C2	5	—	64-bit	Activity Monitors Count Enable Set Register 0
AMCNTENCLR1_EL0	3	3	C13	C3	0	—	64-bit	Activity Monitors Count Enable Clear Register 1

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMCNTENSET1_EL0	3	3	C13	C3	1	—	64-bit	Activity Monitors Count Enable Set Register 1
AMEVCNTR00_EL0	3	3	C13	C4	0	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR01_EL0	3	3	C13	C4	1	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR02_EL0	3	3	C13	C4	2	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR03_EL0	3	3	C13	C4	3	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVTYPER00_EL0	3	3	C13	C6	0	—	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	3	C13	C6	1	—	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	3	C13	C6	2	—	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	3	C13	C6	3	—	64-bit	Activity Monitors Event Type Registers 0
AMEVCNTR10_EL0	3	3	C13	C12	0	—	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR11_EL0	3	3	C13	C12	1	—	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR12_EL0	3	3	C13	C12	2	—	64-bit	Activity Monitors Event Counter Registers 1
AMEVTYPER10_EL0	3	3	C13	C14	0	—	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	3	C13	C14	1	—	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	3	C13	C14	2	—	64-bit	Activity Monitors Event Type Registers 1

21.5 External AMU registers

The 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 21-3: AMU registers summary

Offset	Name	Reset	Width	Description
0x0	AMEVCNTR00[31:0]	—	32-bit	Activity Monitors Event Counter Registers 0
0x4	AMEVCNTR00[63:32]	—	32-bit	Activity Monitors Event Counter Registers 0
0x8	AMEVCNTR01[31:0]	—	32-bit	Activity Monitors Event Counter Registers 0
0xC	AMEVCNTR01[63:32]	—	32-bit	Activity Monitors Event Counter Registers 0
0x10	AMEVCNTR02[31:0]	—	32-bit	Activity Monitors Event Counter Registers 0
0x14	AMEVCNTR02[63:32]	—	32-bit	Activity Monitors Event Counter Registers 0
0x18	AMEVCNTR03[31:0]	—	32-bit	Activity Monitors Event Counter Registers 0
0x1C	AMEVCNTR03[63:32]	—	32-bit	Activity Monitors Event Counter Registers 0
0x100	AMEVCNTR10[31:0]	—	32-bit	Activity Monitors Event Counter Registers 1
0x104	AMEVCNTR10[63:32]	—	32-bit	Activity Monitors Event Counter Registers 1

Offset	Name	Reset	Width	Description
0x108	AMEVCNTR11[31:0]	—	32-bit	Activity Monitors Event Counter Registers 1
0x10C	AMEVCNTR11[63:32]	—	32-bit	Activity Monitors Event Counter Registers 1
0x110	AMEVCNTR12[31:0]	—	32-bit	Activity Monitors Event Counter Registers 1
0x114	AMEVCNTR12[63:32]	—	32-bit	Activity Monitors Event Counter Registers 1
0x400	AMEVTYPER00	—	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	—	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	—	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	—	32-bit	Activity Monitors Event Type Registers 0
0x480	AMEVTYPER10	—	32-bit	Activity Monitors Event Type Registers 1
0x484	AMEVTYPER11	—	32-bit	Activity Monitors Event Type Registers 1
0x488	AMEVTYPER12	—	32-bit	Activity Monitors Event Type Registers 1
0xC00	AMCNTENSET0	—	32-bit	Activity Monitors Count Enable Set Register 0
0xC04	AMCNTENSET1	—	32-bit	Activity Monitors Count Enable Set Register 1
0xC20	AMCNTENCLR0	—	32-bit	Activity Monitors Count Enable Clear Register 0
0xC24	AMCNTENCLR1	—	32-bit	Activity Monitors Count Enable Clear Register 1
0xCE0	AMCGCR	—	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	—	32-bit	Activity Monitors Configuration Register
0xE04	AMCR	—	32-bit	Activity Monitors Control Register
0xE08	AMIIDR	—	32-bit	Activity Monitors Implementation Identification Register
0xFA8	AMDEVAFF0	—	32-bit	Activity Monitors Device Affinity Register 0
0xFAC	AMDEVAFF1	—	32-bit	Activity Monitors Device Affinity Register 1
0xFBC	AMDEVARCH	—	32-bit	Activity Monitors Device Architecture Register
0xFCC	AMDEVTYPE	—	32-bit	Activity Monitors Device Type Register
0xFD0	AMPIDR4	—	32-bit	Activity Monitors Peripheral Identification Register 4
0xFE0	AMPIDR0	—	32-bit	Activity Monitors Peripheral Identification Register 0
0xFE4	AMPIDR1	—	32-bit	Activity Monitors Peripheral Identification Register 1
0xFE8	AMPIDR2	—	32-bit	Activity Monitors Peripheral Identification Register 2
0xFEC	AMPIDR3	—	32-bit	Activity Monitors Peripheral Identification Register 3
0xFF0	AMCIDR0	—	32-bit	Activity Monitors Component Identification Register 0
0xFF4	AMCIDR1	—	32-bit	Activity Monitors Component Identification Register 1
0xFF8	AMCIDR2	—	32-bit	Activity Monitors Component Identification Register 2
0xFFC	AMCIDR3	—	32-bit	Activity Monitors Component Identification Register 3

Appendix A AArch64 registers

This appendix contains the descriptions for the Cortex-A520 AArch64 registers.

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

A.1 AArch64 Generic System Control registers summary

The 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-1: Generic System Control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ACTLR_EL1	3	0	C1	C0	1	–	64-bit	Auxiliary Control Register (EL1)
RGSR_EL1	3	0	C1	C0	5	–	64-bit	Random Allocation Tag Seed Register.
GCR_EL1	3	0	C1	C0	6	–	64-bit	Tag Control Register.
TTBR0_EL1	3	0	C2	C0	0	–	64-bit	Translation Table Base Register 0 (EL1)
TTBR1_EL1	3	0	C2	C0	1	–	64-bit	Translation Table Base Register 1 (EL1)
TCR_EL1	3	0	C2	C0	2	–	64-bit	Translation Control Register (EL1)
APIAKeyLo_EL1	3	0	C2	C1	0	–	64-bit	Pointer Authentication Key A for Instruction (bits[63:0])
APIAKeyHi_EL1	3	0	C2	C1	1	–	64-bit	Pointer Authentication Key A for Instruction (bits[127:64])
APIBKeyLo_EL1	3	0	C2	C1	2	–	64-bit	Pointer Authentication Key B for Instruction (bits[63:0])
APIBKeyHi_EL1	3	0	C2	C1	3	–	64-bit	Pointer Authentication Key B for Instruction (bits[127:64])
APDAKeyLo_EL1	3	0	C2	C2	0	–	64-bit	Pointer Authentication Key A for Data (bits[63:0])
APDAKeyHi_EL1	3	0	C2	C2	1	–	64-bit	Pointer Authentication Key A for Data (bits[127:64])
APDBKeyLo_EL1	3	0	C2	C2	2	–	64-bit	Pointer Authentication Key B for Data (bits[63:0])
APDBKeyHi_EL1	3	0	C2	C2	3	–	64-bit	Pointer Authentication Key B for Data (bits[127:64])
APGAKeyLo_EL1	3	0	C2	C3	0	–	64-bit	Pointer Authentication Key A for Code (bits[63:0])
APGAKeyHi_EL1	3	0	C2	C3	1	–	64-bit	Pointer Authentication Key A for Code (bits[127:64])
SPSel	3	0	C4	C2	0	–	64-bit	Stack Pointer Select
CurrentEL	3	0	C4	C2	2	–	64-bit	Current Exception Level
PAN	3	0	C4	C2	3	–	64-bit	Privileged Access Never
UAO	3	0	C4	C2	4	–	64-bit	User Access Override
AFSR0_EL1	3	0	C5	C1	0	–	64-bit	Auxiliary Fault Status Register 0 (EL1)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AFSR1_EL1	3	0	C5	C1	1	—	64-bit	Auxiliary Fault Status Register 1 (EL1)
ESR_EL1	3	0	C5	C2	0	—	64-bit	Exception Syndrome Register (EL1)
TFSR_EL1	3	0	C5	C6	0	—	64-bit	Tag Fault Status Register (EL1)
TFSRE0_EL1	3	0	C5	C6	1	—	64-bit	Tag Fault Status Register (EL0).
FAR_EL1	3	0	C6	C0	0	—	64-bit	Fault Address Register (EL1)
PAR_EL1	3	0	C7	C4	0	—	64-bit	Physical Address Register
MAIR_EL1	3	0	C10	C2	0	—	64-bit	Memory Attribute Indirection Register (EL1)
AMAIR_EL1	3	0	C10	C3	0	—	64-bit	Auxiliary Memory Attribute Indirection Register (EL1)
LORSA_EL1	3	0	C10	C4	0	—	64-bit	LORegion Start Address (EL1)
LOREA_EL1	3	0	C10	C4	1	—	64-bit	LORegion End Address (EL1)
LORN_EL1	3	0	C10	C4	2	—	64-bit	LORegion Number (EL1)
LORC_EL1	3	0	C10	C4	3	—	64-bit	LORegion Control (EL1)
LORID_EL1	3	0	C10	C4	7	—	64-bit	LORegionID (EL1)
VBAR_EL1	3	0	C12	C0	0	—	64-bit	Vector Base Address Register (EL1)
ISR_EL1	3	0	C12	C1	0	—	64-bit	Interrupt Status Register
CONTEXTIDR_EL1	3	0	C13	C0	1	—	64-bit	Context ID Register (EL1)
TPIDR_EL1	3	0	C13	C0	4	—	64-bit	EL1 Software Thread ID Register
SCXTNUM_EL1	3	0	C13	C0	7	—	64-bit	EL1 Read/Write Software Context Number
IMP_CPUACTLR_EL1	3	0	C15	C1	0	—	64-bit	CPU Auxiliary Control Register
IMP_CPUACTLR2_EL1	3	0	C15	C1	1	—	64-bit	CPU Auxiliary Control Register 2
IMP_CPUACTLR3_EL1	3	0	C15	C1	2	—	64-bit	CPU Auxiliary Control Register 3
IMP_CMPXACTLR_EL1	3	0	C15	C1	3	—	64-bit	Complex Auxiliary Control Register
IMP_CPUECTLR_EL1	3	0	C15	C1	4	—	64-bit	CPU Extended Control Register
IMP_CMPXECTLR_EL1	3	0	C15	C1	7	—	64-bit	Complex Extended Control Register
IMP_CPUPWRCTLR_EL1	3	0	C15	C2	7	—	64-bit	CPU Power Control Register
IMP_ATCR_EL1	3	0	C15	C7	0	—	64-bit	CPU Auxiliary Translation Control Register
AIDR_EL1	3	1	C0	C0	7	—	64-bit	Auxiliary ID Register
NZCV	3	3	C4	C2	0	—	64-bit	Condition Flags
DAIF	3	3	C4	C2	1	—	64-bit	Interrupt Mask Bits
DIT	3	3	C4	C2	5	—	64-bit	Data Independent Timing
SSBS	3	3	C4	C2	6	—	64-bit	Speculative Store Bypass Safe
TCO	3	3	C4	C2	7	—	64-bit	Tag Check Override
FPCR	3	3	C4	C4	0	—	64-bit	Floating-point Control Register
FPSR	3	3	C4	C4	1	—	64-bit	Floating-point Status Register
TPIDR_EL0	3	3	C13	C0	2	—	64-bit	EL0 Read/Write Software Thread ID Register
TPIDRRO_EL0	3	3	C13	C0	3	—	64-bit	EL0 Read-Only Software Thread ID Register
SCXTNUM_EL0	3	3	C13	C0	7	—	64-bit	EL0 Read/Write Software Context Number
ACTLR_EL2	3	4	C1	C0	1	—	64-bit	Auxiliary Control Register (EL2)
HACR_EL2	3	4	C1	C1	7	—	64-bit	Hypervisor Auxiliary Control Register
TTBR0_EL2	3	4	C2	C0	0	—	64-bit	Translation Table Base Register 0 (EL2)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TTBR1_EL2	3	4	C2	C0	1	–	64-bit	Translation Table Base Register 1 (EL2)
TCR_EL2	3	4	C2	C0	2	–	64-bit	Translation Control Register (EL2)
VTTBR_EL2	3	4	C2	C1	0	–	64-bit	Virtualization Translation Table Base Register
VTCR_EL2	3	4	C2	C1	2	–	64-bit	Virtualization Translation Control Register
VSTTBR_EL2	3	4	C2	C6	0	–	64-bit	Virtualization Secure Translation Table Base Register
VSTCR_EL2	3	4	C2	C6	2	–	64-bit	Virtualization Secure Translation Control Register
AFSR0_EL2	3	4	C5	C1	0	–	64-bit	Auxiliary Fault Status Register 0 (EL2)
AFSR1_EL2	3	4	C5	C1	1	–	64-bit	Auxiliary Fault Status Register 1 (EL2)
ESR_EL2	3	4	C5	C2	0	–	64-bit	Exception Syndrome Register (EL2)
TFSR_EL2	3	4	C5	C6	0	–	64-bit	Tag Fault Status Register (EL2)
FAR_EL2	3	4	C6	C0	0	–	64-bit	Fault Address Register (EL2)
HPFAR_EL2	3	4	C6	C0	4	–	64-bit	Hypervisor IPA Fault Address Register
MAIR_EL2	3	4	C10	C2	0	–	64-bit	Memory Attribute Indirection Register (EL2)
AMAIR_EL2	3	4	C10	C3	0	–	64-bit	Auxiliary Memory Attribute Indirection Register (EL2)
VBAR_EL2	3	4	C12	C0	0	–	64-bit	Vector Base Address Register (EL2)
CONTEXTIDR_EL2	3	4	C13	C0	1	–	64-bit	Context ID Register (EL2)
TPIDR_EL2	3	4	C13	C0	2	–	64-bit	EL2 Software Thread ID Register
SCXTNUM_EL2	3	4	C13	C0	7	–	64-bit	EL2 Read/Write Software Context Number
IMP_ATCR_EL2	3	4	C15	C7	0	–	64-bit	CPU Auxiliary Translation Control Register
IMP_AVTCR_EL2	3	4	C15	C7	1	–	64-bit	CPU Auxiliary Translation Control Register
ACTLR_EL3	3	6	C1	C0	1	–	64-bit	Auxiliary Control Register (EL3)
SCR_EL3	3	6	C1	C1	0	–	64-bit	Secure Configuration Register
CPTR_EL3	3	6	C1	C1	2	–	64-bit	Architectural Feature Trap Register (EL3)
MDCR_EL3	3	6	C1	C3	1	–	64-bit	Monitor Debug Configuration Register (EL3)
TTBR0_EL3	3	6	C2	C0	0	–	64-bit	Translation Table Base Register 0 (EL3)
TCR_EL3	3	6	C2	C0	2	–	64-bit	Translation Control Register (EL3)
AFSR0_EL3	3	6	C5	C1	0	–	64-bit	Auxiliary Fault Status Register 0 (EL3)
AFSR1_EL3	3	6	C5	C1	1	–	64-bit	Auxiliary Fault Status Register 1 (EL3)
ESR_EL3	3	6	C5	C2	0	–	64-bit	Exception Syndrome Register (EL3)
TFSR_EL3	3	6	C5	C6	0	–	64-bit	Tag Fault Status Register (EL3)
FAR_EL3	3	6	C6	C0	0	–	64-bit	Fault Address Register (EL3)
MAIR_EL3	3	6	C10	C2	0	–	64-bit	Memory Attribute Indirection Register (EL3)
AMAIR_EL3	3	6	C10	C3	0	–	64-bit	Auxiliary Memory Attribute Indirection Register (EL3)
VBAR_EL3	3	6	C12	C0	0	–	64-bit	Vector Base Address Register (EL3)
RVBAR_EL3	3	6	C12	C0	1	–	64-bit	Reset Vector Base Address Register (if EL3 implemented)
RMR_EL3	3	6	C12	C0	2	–	64-bit	Reset Management Register (EL3)
TPIDR_EL3	3	6	C13	C0	2	–	64-bit	EL3 Software Thread ID Register
SCXTNUM_EL3	3	6	C13	C0	7	–	64-bit	EL3 Read/Write Software Context Number
IMP_ATCR_EL3	3	6	C15	C7	0	–	64-bit	CPU Auxiliary Translation Control Register

A.1.1 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 AArch64-HCR_EL2.{E2H, TGE} is {1, 1}, and instead the configuration and control fields are provided by the AArch64-ACTLR_EL2 register. This avoids the need for software to manage the contents of these register when switching between a Guest OS and a Host OS.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Figure A-1: AArch64_actlr_el1 bit assignments

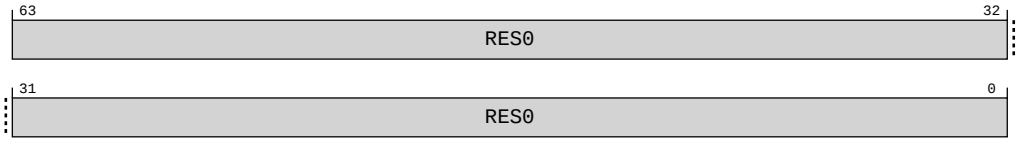


Table A-2: ACTLR_EL1 bit descriptions

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

MRS <Xt>, ACTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ACTLR_EL1;
elsif PSTATE.EL == EL2 then
    return ACTLR_EL1;
elsif PSTATE.EL == EL3 then
    return ACTLR_EL1;
```

MSR ACTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ACTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    ACTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ACTLR_EL1 = X[t];
```

A.1.2 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

See bit descriptions

Reset value

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

Figure A-2: AArch64_afsr0_el1 bit assignments

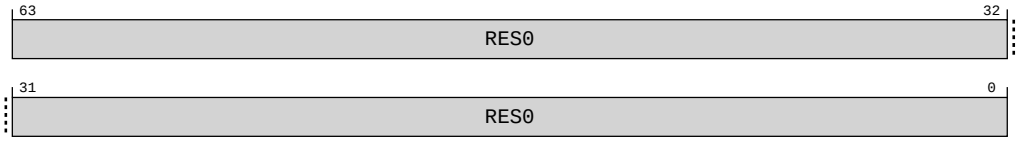


Table A-5: AFSR0_EL1 bit descriptions

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

Access

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic AFSR0_EL1 or AFSR0_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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 AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic AFSR0_EL1 or AFSR0_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.
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);
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR0_EL2;
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    return AFSR0_EL1;
```

MSR AFSR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL2 = X[t];
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t];
```

MRS <Xt>, AFSR0_EL12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR0_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
```

```
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        return AFSR0_EL1;
    else
        UNDEFINED;
```

MSR AFSR0_EL12, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        AFSR0_EL1 = X[t];
    else
        UNDEFINED;
```

A.1.3 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

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-3: AArch64_afsr1_el1 bit assignments

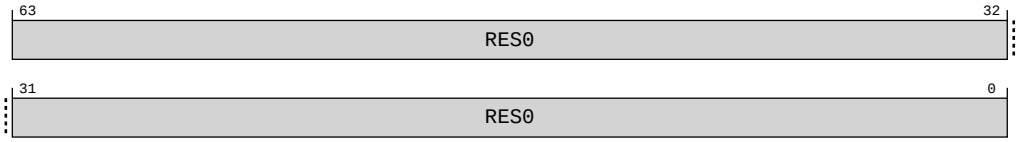


Table A-10: AFSR1_EL1 bit descriptions

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

Access

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic AFSR1_EL1 or AFSR1_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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 AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic AFSR1_EL1 or AFSR1_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.
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);
    else
        return AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR1_EL2;
    else
        return AFSR1_EL1;
elsif PSTATE.EL == EL3 then
    return AFSR1_EL1;
```

MSR AFSR1_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AFSR1_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL2 = X[t];
    else
        AFSR1_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR1_EL1 = X[t];
```

MRS <Xt>, AFSR1_EL12

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

MSR AFSR1_EL12, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
```

```
if EL2Enabled() && HCR_EL2.E2H == '1' then
    AFSR1_EL1 = X[t];
else
    UNDEFINED;
```

A.1.4 PAR_EL1, Physical Address Register

Returns the output address (OA) from an Address translation instruction that executed successfully, or fault information if the instruction did not execute successfully.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When AArch64-PAR_EL1.F == '0'

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

When AArch64-PAR_EL1.F == '1'

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

Where the reset reads xxxx, see individual bits

Bit descriptions

When AArch64-PAR_EL1.F == '0'

This section describes the register value returned by the successful execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

On a successful conversion, the PAR_EL1 can return a value that indicates the resulting attributes, rather than the values that appear in the translation table descriptors. More precisely:

- The PAR_EL1.{ATTR, SH} fields are permitted to report the resulting attributes, as determined by any permitted implementation choices and any applicable configuration bits, instead of reporting the values that appear in the translation table descriptors.

- See the PAR_EL1.NS bit description for constraints on the value it returns.

Figure A-4: AArch64_par_el1 bit assignments

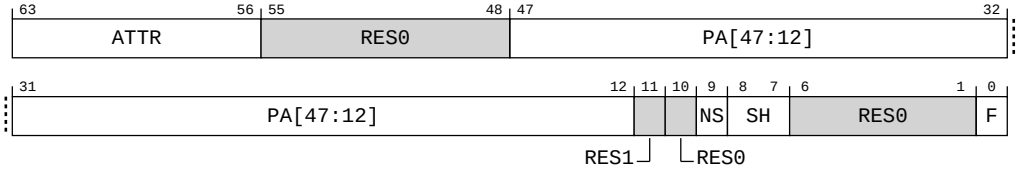


Table A-15: PAR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	ATTR	Memory attributes for the returned output address. This field uses the same encoding as the Attr<n> fields in AArch64-MAIR_EL1, AArch64-MAIR_EL2, and AArch64-MAIR_EL3. The value returned in this field can be the resulting attribute that is actually implemented by the implementation, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor. **Note:** The attributes presented are consistent with the stages of translation applied in the address translation instruction. If the instruction performed a stage 1 translation only, the attributes are from the stage 1 translation. If the instruction performed a stage 1 and stage 2 translation, the attributes are from the combined stage 1 and stage 2 translation.	8{x}
[55:48]	**RES0**	Reserved	**RES0**
[47:12]	PA[47:12]	Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[47:12]. For implementations with fewer than 48 physical address bits, the corresponding upper bits in this field are **RES0**.	36{x}
[11]	**RES1**	Reserved	**RES1**
[10]	**RES0**	Reserved	**RES0**
[9]	NS	Non-secure. The NS attribute for a translation table entry from a Secure translation regime. For a result from a Secure translation regime, when AArch64-SCR_EL3.EEL2 is 1, this bit reflects the Security state of the intermediate physical address space of the translation for the instructions: • In AArch64 state: AT S1E1R, AT S1E1W, AT S1E1RP, AT S1E1WP, AT S1E0R, and AT S1E0W. Otherwise, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation. For a result from a Non-secure translation regime, this bit is **UNKNOWN**.	x

Bits	Name	Description	Reset
[8:7]	SH	Shareability attribute, for the returned output address. **0b00** Non-shareable. **0b10** Outer Shareable. **0b11** Inner Shareable. The value 0b01 is reserved. **Note:** This field returns the value 0b10 for: • Any type of Device memory. • Normal memory with both Inner Non-cacheable and Outer Non-cacheable attributes. The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.	xx
[6:1]	**RES0**	Reserved	**RES0**
[0]	F	Indicates whether the instruction performed a successful address translation. **0b0** Address translation completed successfully.	x

When AArch64-PAR_EL1.F == '1'

This section describes the register value returned by a fault on the execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

Figure A-5: AArch64_par_el1 bit assignments

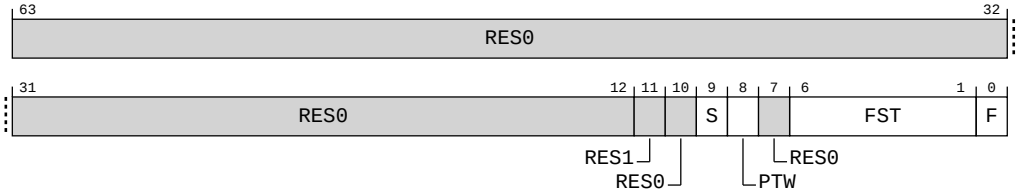


Table A-16: PAR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:12]	**RES0**	Reserved	**RES0**
[11]	**RES1**	Reserved	**RES1**
[10]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[9]	S	Indicates the translation stage at which the translation aborted: **0b0** Translation aborted because of a fault in the stage 1 translation. **0b1** Translation aborted because of a fault in the stage 2 translation.	x
[8]	PTW	If this bit is set to 1, it indicates the translation aborted because of a stage 2 fault during a stage 1 translation table walk.	x
[7]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[6:1]	FST	Fault status code, as shown in the Data Abort ESR encoding. **0b000000** Address size fault, level 0 of translation or translation table base register. **0b000001** Address size fault, level 1. **0b000010** Address size fault, level 2. **0b000011** Address size fault, level 3. **0b000100** Translation fault, level 0. **0b000101** Translation fault, level 1. **0b000110** Translation fault, level 2. **0b000111** Translation fault, level 3. **0b001001** Access flag fault, level 1. **0b001010** Access flag fault, level 2. **0b001011** Access flag fault, level 3. **0b001101** Permission fault, level 1. **0b001110** Permission fault, level 2. **0b001111** Permission fault, level 3. **0b010100** Synchronous External abort on translation table walk or hardware update of translation table, level 0. **0b010101** Synchronous External abort on translation table walk or hardware update of translation table, level 1. **0b010110** Synchronous External abort on translation table walk or hardware update of translation table, level 2. **0b010111** Synchronous External abort on translation table walk or hardware update of translation table, level 3. **0b110000** TLB conflict abort. **0b110001** Unsupported atomic hardware update fault.	6{x}

Bits	Name	Description	Reset
[0]	F	Indicates whether the instruction performed a successful address translation. **0b1** Address translation aborted.	x

Access

MRS <Xt>, PAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0111	0b0100	0b000

MSR PAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0111	0b0100	0b000

Accessibility

MRS <Xt>, PAR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return PAR_EL1;
elsif PSTATE.EL == EL2 then
    return PAR_EL1;
elsif PSTATE.EL == EL3 then
    return PAR_EL1;
```

MSR PAR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PAR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    PAR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    PAR_EL1 = X[t];
```

A.1.5 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register (EL1)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by AArch64-MAIR_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

AMAIR_EL1 is permitted to be cached in a TLB.

Figure A-6: AArch64_amair_el1 bit assignments

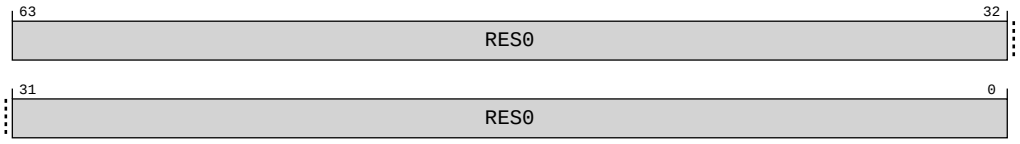


Table A-19: AMAIR_EL1 bit descriptions

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

Access

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic AMAIR_EL1 or AMAIR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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 AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic AMAIR_EL1 or AMAIR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.
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);
    else
        return AMAIR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AMAIR_EL2;
    else
        return AMAIR_EL1;
elsif PSTATE.EL == EL3 then
    return AMAIR_EL1;
```

MSR AMAIR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AMAIR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
```

```
    if HCR_EL2.E2H == '1' then
        AMAIR_EL2 = X[t];
    else
        AMAIR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AMAIR_EL1 = X[t];
```

MRS <Xt>, AMAIR_EL12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AMAIR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        return AMAIR_EL1;
    else
        UNDEFINED;
```

MSR AMAIR_EL12, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t];
    else
        UNDEFINED;
```

A.1.6 LORID_EL1, LORegionID (EL1)

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

Configurations

If no LORegion descriptors are implemented, then the registers AArch64-LORC_EL1, AArch64-LORN_EL1, AArch64-LOREA_EL1, and AArch64-LORSA_EL1 are RES0.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Figure A-7: AArch64_lorid_el1 bit assignments

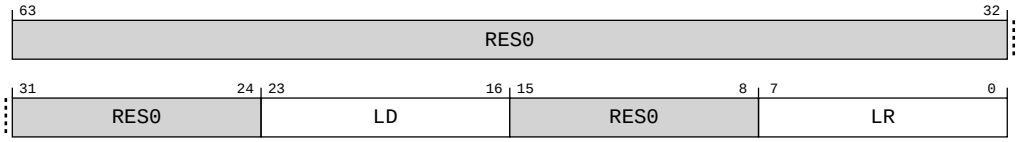


Table A-24: LORID_EL1 bit descriptions

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. **0b00000100** Four LOR descriptors are supported	8{x}
[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. **0b00000100** Four LORegions are supported	8{x}

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return LORID_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return LORID_EL1;
elsif PSTATE.EL == EL3 then
    return LORID_EL1;
```

A.1.7 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Figure A-8: AArch64_imp_cpuactlr_el1 bit assignments

Table A-26: 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
        return IMP_CPUACTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUACTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUACTLR_EL1;
```

MSR S3_0_C15_C1_0, <Xt>

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

```
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ACTLREN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ACTLREN == '0' then
        UNDEFINED;
    elsif ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR_EL1 = X[t];
```

A.1.8 IMP_CPUACTLR2_EL1, CPU Auxiliary Control Register 2

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-9: AArch64_imp_cpuactlr2_el1 bit assignments

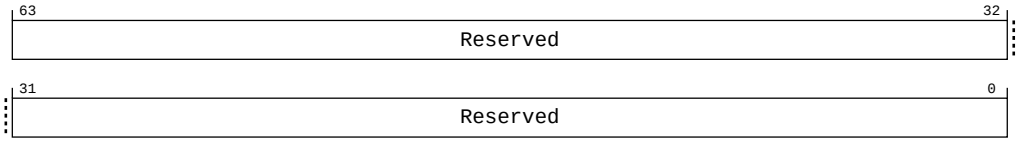


Table A-29: IMP_CPUACTLR2_EL1 bit descriptions

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

MSR S3_0_C15_C1_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ACTLREN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ACTLREN == '0' then
        UNDEFINED;
    elsif ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR2_EL1 = X[t];
```

A.1.9 IMP_CPUACTLR3_EL1, CPU Auxiliary Control Register 3

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

See bit descriptions

Reset value

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

Figure A-10: AArch64_imp_cpuactlr3_el1 bit assignments

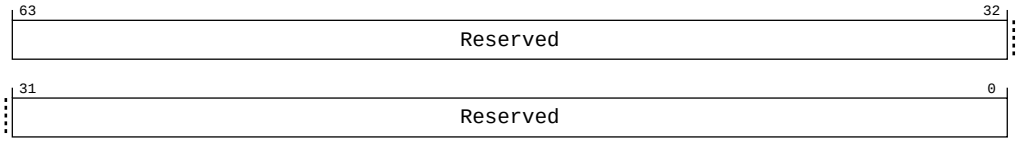


Table A-32: IMP_CPUACTLR3_EL1 bit descriptions

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
        return IMP_CPUACTLR3_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUACTLR3_EL1;
elsif PSTATE.EL == EL3 then
    return 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 Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ACTLREN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR3_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ACTLREN == '0' then
        UNDEFINED;
    elsif ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        IMP_CPUACTLR3_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR3_EL1 = X[t];
```

A.1.10 IMP_CMPXACTLR_EL1, Complex Auxiliary Control Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Figure A-11: AArch64_imp_cmpxactlr_el1 bit assignments

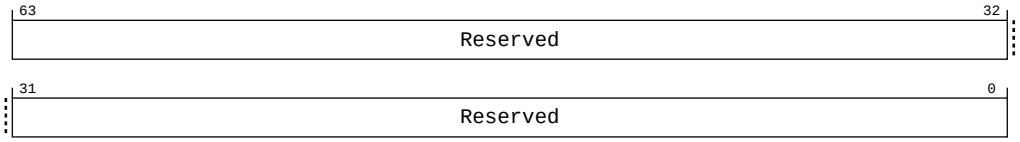


Table A-35: IMP_CMPXACTLR_EL1 bit descriptions

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
        return IMP_CMPXACTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CMPXACTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CMPXACTLR_EL1;
```

MSR S3_0_C15_C1_3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ACTLREN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMPXACTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ACTLREN == '0' then
        UNDEFINED;
    elsif ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMPXACTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CMPXACTLR_EL1 = X[t];
```

A.1.11 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

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx 00xx xxx0 00xx xxx0 0000 0000 1x01 xxx0 0000 000x 00xx xxx0

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-12: AArch64_imp_cpuectlr_el1 bit assignments

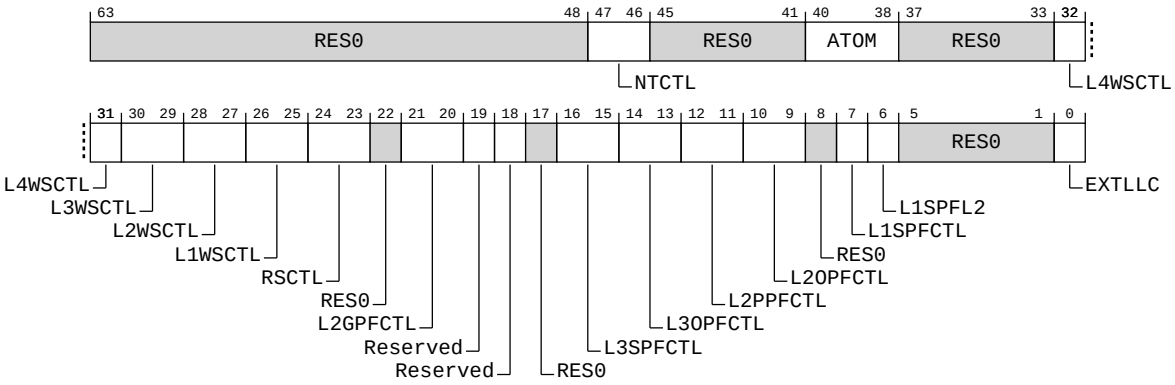


Table A-38: IMP_CPUECTLR_EL1 bit descriptions

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

Bits	Name	Description	Reset
[47:46]	NTCTL	Transient/non-temporal L1 eviction control. **0b00** Transient/non-temporal lines evicted from the L1 cache skip L2 allocation, and allocate into the L3 cache as least-recently-used. **0b01** Transient/non-temporal lines evicted from the L1 cache allocate to the L2 as least-recently-used, and when evicted from the L2 allocate to the L3 as near-least-recently-used. **0b10** Transient/non-temporal clean lines evicted from the L1 cache are evicted without data. Dirty lines skip L2 allocation. Their allocation and replacement policy at L3 depends on the current RSCTL setting. **0b11** Transient/non-temporal lines evicted from the L1 cache skip L2 allocation, and allocate into the L3 cache as near-least-recently-used.	0b00
[45:41]	**RES0**	Reserved	**RES0**
[40:38]	ATOM	Atomic instruction handling policy **0b000** Atomic stores will be executed far unless they hit in a unique state in the L1 data cache, all other atomic instructions will be executed near. **0b001** All atomic instructions will be executed far unless they hit in a unique state in the L1 data cache. **0b010** All atomic instructions will be executed near. **0b011** All atomic instructions will be executed far. **0b100** Atomic stores will be executed far unless they hit in a unique state in the L1 data cache, all other atomic instructions will be executed near if they hit the L1 data cache, far otherwise.	0b000
[37:33]	**RES0**	Reserved	**RES0**
[32:31]	L4WSCTL	System cache write streaming threshold. Controls the threshold of the number of consecutive cache lines which are fully written without being read before stores are marked as Outer No Write-Allocate. **0b00** 512 cache lines. **0b01** 2048 cache lines. **0b10** 8191 cache lines. **0b11** Disable write streaming through system cache. All cache lines fetched due to stores will be marked as Outer Write-Allocate.	0b00

Bits	Name	Description	Reset
[30:29]	L3WSCTL	L3 write streaming threshold. Controls the threshold of the number of consecutive cache lines which are fully written without being read before stores stop causing L3 cache allocations. **0b00** 128 cache lines. **0b01** 1024 cache lines. **0b10** 4096 cache lines. **0b11** Disable write streaming through L3 cache. All cache lines fetched due to stores will allocate in L1, L2 or L3 caches.	0b00
[28:27]	L2WSCTL	L2 write streaming threshold. Controls the threshold of the number of consecutive cache lines which are fully written without being read before stores stop causing L2 cache allocations. **0b00** 16 cache lines. **0b01** 128 cache lines. **0b10** 512 cache lines. **0b11** Disable write streaming through L2 cache. All cache lines fetched due to stores will allocate in L1 or L2 caches.	0b00
[26:25]	L1WSCTL	L1 write streaming threshold. Controls the threshold of the number of consecutive cache lines which are fully written without being read before stores stop causing L1 cache allocations. **0b00** 4 cache lines. **0b01** 64 cache lines. **0b10** 128 cache lines. **0b11** Disable write streaming.	0b00
[24:23]	RSCTL	Read streaming aggressiveness control. **0b00** Enabled. Some dataless evictions may occur. **0b01** Enabled, more conservative. Some dataless evictions may occur. **0b10** Enabled, most conservative. No dataless evictions occur. **0b11** Read streaming disabled.	0b01
[22]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[21:20]	L2GPFCTL	L2 cache spatial prefetcher aggressiveness control. **0b01** Aggressive L2 spatial prefetching. **0b10** Conservative L2 spatial prefetching. **0b11** Very Conservative L2 spatial prefetching.	0b01
[19]	Reserved_19	Reserved for Arm internal use	x
[18]	Reserved_18	Reserved for Arm internal use	x
[17]	**RES0**	Reserved	**RES0**
[16:15]	L3SPFCTL	L3 cache stride prefetcher aggressiveness control. **0b00** Dynamic L3 stride prefetcher aggressiveness. **0b01** Conservative L3 stride prefetching. **0b10** Aggressive L3 stride prefetching.	0b00
[14:13]	L3OPFCTL	L3 cache offset prefetcher aggressiveness control. **0b00** Dynamic L3 offset prefetcher aggressiveness. **0b01** Conservative L3 offset prefetching. **0b10** Aggressive L3 offset prefetching. **0b11** L3 offset prefetching disabled.	0b00
[12:11]	L2PPFCTL	L2 cache pattern prefetcher aggressiveness control. **0b00** Very conservative L2 pattern prefetching. **0b01** Conservative L2 pattern prefetching. **0b11** Aggressive L2 pattern prefetching.	0b00
[10:9]	L2OPFCTL	L2 cache offset prefetcher aggressiveness control. **0b00** Dynamic L2 offset prefetcher aggressiveness. **0b01** Conservative L2 offset prefetching. **0b10** Very conservative L2 offset prefetching. **0b11** Most conservative L2 offset prefetching.	0b00

Bits	Name	Description	Reset
[8]	RES0	Reserved	RES0
[7]	L1SPFCTL	L1 cache stride prefetcher agressiveness control. **0b0** Dynamic stride prefetcher aggressiveness. **0b1** Conservative stride prefetching.	0b0
[6]	L1SPFL2	Stride prefetcher cache level control. **0b0** Stride prefetcher prefetches into L1 and L3. **0b1** Stride prefetcher prefetches into L1 and L2.	0b0
[5:1]	RES0	Reserved	RES0
[0]	EXTLLC	Indicates that an external Last-level cache is present in the system, and that the DataSource field on the master CHI interface will indicate when data is returned from the LLC. Used to control how the LL_CACHE* PMU events count. **0b0** The last level cache in PMU events is within the cluster. **0b1** The last level cache in PMU events is outside the cluster.	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
        return IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUECTLR_EL1;
```

MSR S3_0_C15_C1_4, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ECTLREN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ECTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ECTLREN == '0' then
        UNDEFINED;
    elsif ACTLR_EL3.ECTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUECTLR_EL1 = X[t];
```

A.1.12 IMP_CMPXECTLR_EL1, Complex Extended Control Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx 0100 0000 xxxx xxxx x0xx xxxx xxxx 1001 0100 xx00 x000


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-13: AArch64_imp_cmpxectlr_el1 bit assignments

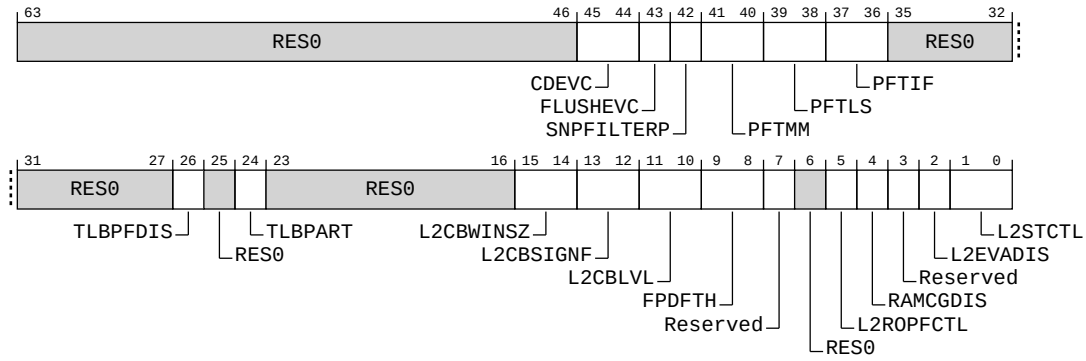


Table A-41: IMP_CMPXECTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:46]	**RES0**	Reserved	**RES0**
[45:44]	CDEVC	Downstream Cache Control **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.	xx
[43]	FLUSHEVC	Eviction Flush Control **0b0** Disables sending data when hardware cache flushes or DC CISW instructions evict a clean cache-line. . **0b1** Sending of data when hardware cache flushes or DC CISW instructions evict clean cache-lines is controlled by Downstream Cache Control. Sending of Evict transactions is controlled by SNPFILTERP.	0b0
[42]	SNPFILTERP	Downstream Snoop Filter Present **0b0** Disables sending Evict transactions when clean cache-lines are evicted without data. **0b1** Enables sending Evict transactions when clean cache-lines are evicted without data.	0b1

Bits	Name	Description	Reset
[41:40]	PFTMM	DRAM prefetch using PrefetchTgt transactions for table walk requests. **0b00** Disable PrefetchTgt generation for requests from the Memory Management unit (MMU). **0b01** Dynamically generate PrefetchTgt for requests from the MMU. **0b11** Always generate PrefetchTgt for requests from the MMU.	0b00
[39:38]	PFTLS	DRAM prefetch using PrefetchTgt transactions for load and store requests. **0b00** Disable PrefetchTgt generation for requests from the Load-Store unit (LS). **0b01** Dynamically generate PrefetchTgt for requests from the LS. **0b11** Always generate PrefetchTgt for requests from the LS.	0b00
[37:36]	PFTIF	DRAM prefetch using PrefetchTgt transactions for instruction fetch requests. **0b00** Disable PrefetchTgt generation for requests from the Instruction Fetch unit (IF). **0b01** Dynamically generate PrefetchTgt for requests from the IF. **0b11** Always generate PrefetchTgt for requests from the IF.	0b00
[35:27]	**RES0**	Reserved	**RES0**
[26]	TLBPFDIS	Disable L2 TLB prefetcher **0b0** The L2 TLB prefetcher is enabled. **0b1** The L2 TLB prefetcher is disabled.	0b0
[25]	**RES0**	Reserved	**RES0**
[24]	TLBPART	**When the complex contains two cores** Partition L2 TLB allocations by core **0b0** Both cores are able to allocated to the full L2 TLB. **0b1** Core 0 can only allocate to ways 0-3 in the L2 TLB, and core 1 can only allocate to ways 4-7. Cores can hit entries allocated by either core. **Otherwise** RES0	x
[23:16]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[15:14]	L2CBWINSZ	Number of CBUSY responses in one sampling window. **0b00** 64 CBUSY responses per sampling window. **0b01** 128 CBUSY responses per sampling window. **0b10** 256 CBUSY responses per sampling window. **0b11** 512 CBUSY responses per sampling window.	0b10
[13:12]	L2CBSIGNF	Fraction of CBUSY responses in the sampling window necessary to be considered a valid sample of that CBUSY value. **0b00** 1/32 **0b01** 1/16 **0b10** 1/8 **0b11** 1/4	0b01
[11:10]	L2CBLVL	L2 internal CBUSY generation control. **0b00** Disable internal CBUSY generation. **0b01** Normal thresholds. **0b10** Conservative thresholds - throttles early. **0b11** Most conservative thresholds - throttles earlier.	0b01
[9:8]	FPDFTH	Prefetch data forwarding threshold. The value 0b11 disables prefetch data forwarding. **0b00** Default prefetch forwarding behaviour. **0b01** Faster prefetch forwarding timeout. **0b10** Immediate prefetch forwarding timeout (no waiting). **0b11** Prefetch forwarding is disabled.	0b00
[7]	Reserved_7	Reserved for Arm internal use	x
[6]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[5]	L2ROPFCTL	L2 ReadOnce hitting prefetched line age control **0b0** ReadOnce from TLB hitting TLB-prefetched line sets the age to allocation age. **0b1** ReadOnce from TLB hitting TLB-prefetched line sets the age to MRU.	0b0
[4]	RAMCGDIS	Disable clock gating for all RAMs in the complex other than the L2 data RAMs **0b0** Clock gating for all RAMs in the complex other than the L2 data RAMs are enabled. **0b1** Clock gating for all RAMs in the complex other than the L2 data RAMs are disabled.	0b0
[3]	Reserved_3	Reserved for Arm internal use	x
[2]	L2EVADIS	Disable L2 cache data RAM EVA accesses **0b0** Optimized evict/allocate accesses to L2 cache data RAMs using RAM EVA feature are enabled. **0b1** Optimized evict/allocate accesses to L2 cache data RAMs using RAM EVA feature are disabled.	0b0
[1:0]	L2STCTL	L2 cache stashing control **0b00** Stashes targeting L2 cache will allocate as if the line were brought in by a load. **0b01** Stashes targeting L2 cache will allocate and be marked as preferred targets for eviction. **0b10** Stashes targeting L2 cache will allocate as if the line were brought in by a load, but will only allocate to odd numbered cache ways. **0b11** Stashes targeting L2 cache will be ignored.	0b00

Access

MRS <Xt>, S3_0_C15_C1_7

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b111

MSR S3_0_C15_C1_7, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b111

Accessibility

MRS <Xt>, S3_0_C15_C1_7

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

```
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CMPXECTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CMPXECTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CMPXECTLR_EL1;
```

MSR S3_0_C15_C1_7, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ECTLREN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ECTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMPXECTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ECTLREN == '0' then
        UNDEFINED;
    elsif ACTLR_EL3.ECTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMPXECTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CMPXECTLR_EL1 = X[t];
```

A.1.13 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

See bit descriptions

Reset value

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

Figure A-14: AArch64_imp_cpupwrctlr_el1 bit assignments

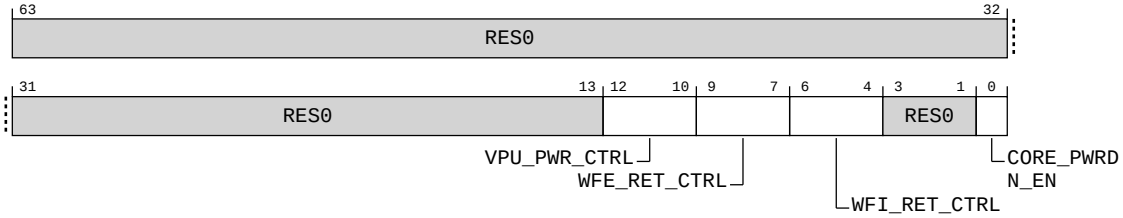


Table A-44: IMP_CPUPWRCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:13]	**RES0**	Reserved	**RES0**
[12:10]	VPU_PWR_CTRL	VPU powerdown control. **0b000** VPU powerdown is disabled. **0b001** 2 system counter ticks are required before VPU powerdown. **0b010** 8 system counter ticks are required before VPU powerdown. **0b011** 32 system counter ticks are required before VPU powerdown. **0b100** 64 system counter ticks are required before VPU powerdown. **0b101** 128 system counter ticks are required before VPU powerdown. **0b110** 256 system counter ticks are required before VPU powerdown. **0b111** 512 system counter ticks are required before VPU powerdown.	xxx

Bits	Name	Description	Reset
[9:7]	WFE_RET_CTRL	Wait for Event retention control. **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.	xxx
[6:4]	WFI_RET_CTRL	Wait for Interrupt retention control. **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.	xxx
[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.	x

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
        return IMP_CPUPWRCTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUPWRCTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPWRCTLR_EL1;
```

MSR S3_0_C15_C2_7, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif Halted() && EDSCR.SDD == '1' && ACTLR_EL3.PWREN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ACTLR_EL2.PWREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.PWREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPWRCTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && ACTLR_EL3.PWREN == '0' then
        UNDEFINED;
    elsif ACTLR_EL3.PWREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPWRCTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUPWRCTLR_EL1 = X[t];
```

A.1.14 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-15: AArch64_imp_atcr_el1 bit assignments

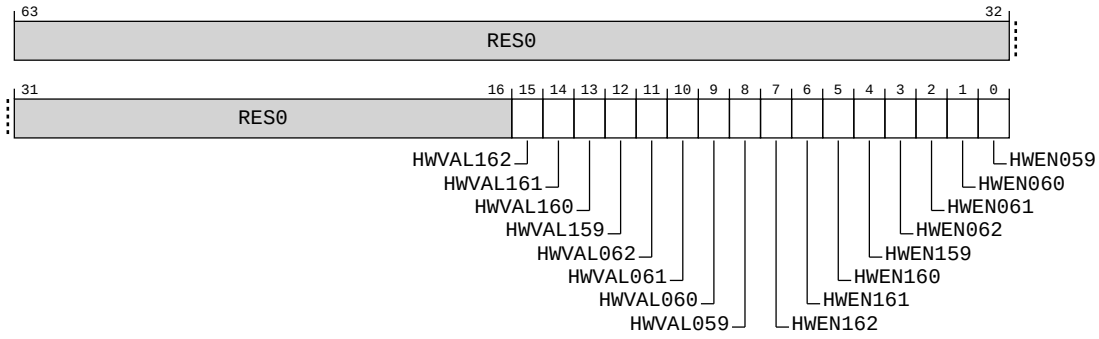


Table A-47: IMP_ATCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to page table walks using TTBR1_EL1 if HWEN162 is set.	x
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to page table walks using TTBR1_EL1 if HWEN161 is set.	x

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

MRS <Xt>, S3_5_C15_C7_0

op0	op1	CRn	CRm	op2
0b11	0b101	0b1111	0b0111	0b000

MSR S3_5_C15_C7_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1111	0b0111	0b000

Accessibility

MRS <Xt>, S3_0_C15_C7_0

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

MSR S3_0_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_ATCR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    IMP_ATCR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL1 = X[t];
```

MRS <Xt>, S3_5_C15_C7_0

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

MSR S3_5_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        IMP_ATCR_EL1 = X[t];
```

```
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        IMP_ATCR_EL1 = X[t];
    else
        UNDEFINED;
```

A.1.15 AIDR_EL1, Auxiliary ID Register

Provides **IMPLEMENTATION DEFINED** identification information.

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Figure A-16: AArch64_aidr_el1 bit assignments

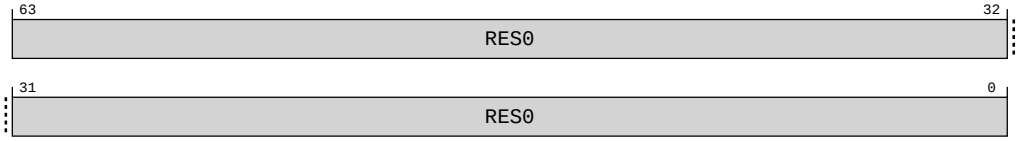


Table A-52: AIDR_EL1 bit descriptions

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
        return AIDR_EL1;
elsif PSTATE.EL == EL2 then
    return AIDR_EL1;
elsif PSTATE.EL == EL3 then
    return AIDR_EL1;
```

A.1.16 ACTLR_EL2, Auxiliary Control Register (EL2)

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


Note

Arm recommends the contents of this register are updated to apply to EL0 when AArch64-HCR_EL2.{E2H, TGE} is {1, 1}, gaining configuration and control fields from the AArch64-ACTLR_EL1. This avoids the need for software to manage the contents of these register when switching between a Guest OS and a Host OS.

Configurations

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

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 0xxx 0xxx xx00

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-17: AArch64_actlr_el2 bit assignments

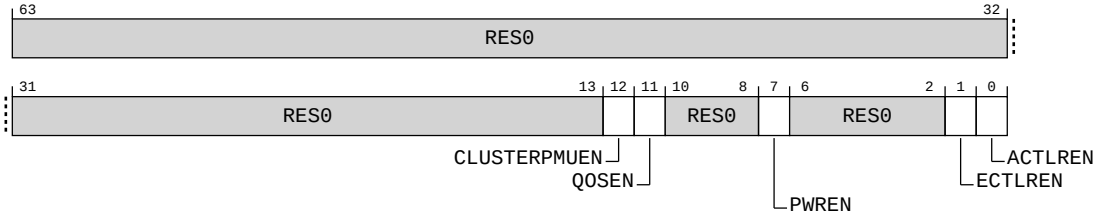


Table A-54: ACTLR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:13]	**RES0**	Reserved	**RES0**
[12]	CLUSTERPMUEN	Cluster PMU Registers enable. Traps EL1 writes to implementation-defined cluster PMU registers to EL2. Possible values of this bit are: **0b0** This control causes writes to IMP_CLUSTERPM* at EL1 to be trapped. **0b1** This control does not cause any instructions to be trapped.	0b0
[11]	QOSEN	Cluster Bus QoS Registers enable. Traps EL1 writes to AArch64-IMP_CLUSTERBUSQOS_EL1 to EL2. Possible values of this bit are: **0b0** This control causes writes to AArch64-IMP_CLUSTERBUSQOS_EL1 at EL1 to be trapped. **0b1** This control does not cause any instructions to be trapped.	0b0
[10:8]	**RES0**	Reserved	**RES0**
[7]	PWREN	Power Control Registers enable. Traps EL1 writes to implementation-defined power control registers to EL2. Possible values of this bit are: **0b0** This control causes writes to AArch64-IMP_CPUPWRCTLR_EL1, AArch64-IMP_CLUSTERPWRCTLR_EL1, AArch64-IMP_CLUSTERPWRDN_EL1 and IMP_CLUSTERL3*_EL1 at EL1 to be trapped. **0b1** This control does not cause any instructions to be trapped.	0b0
[6:2]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[1]	ECTLREN	Extended Control Registers enable. Traps EL1 writes to AArch64-IMP_CPUECTLR_EL1, AArch64-IMP_CMPXECTLR_EL1 and AArch64-IMP_CLUSTERECTLR_EL1 to EL2. Possible values of this bit are: **0b0** This control causes writes to AArch64-IMP_CPUECTLR_EL1, AArch64-IMP_CMPXECTLR_EL1 and AArch64-IMP_CLUSTERECTLR_EL1 at EL1 to be trapped. **0b1** This control does not cause any instructions to be trapped.	0b0
[0]	ACTLREN	Auxiliary Control Registers enable. Traps EL1 writes to AArch64-IMP_CPUACTLR_EL1, AArch64-IMP_CPUACTLR2_EL1, AArch64-IMP_CMPXACTLR_EL1 and AArch64-IMP_CLUSTERACTLR_EL1 to EL2. Possible values of this bit are: **0b0** This control causes writes to AArch64-IMP_CPUACTLR_EL1, AArch64-IMP_CPUACTLR2_EL1, AArch64-IMP_CMPXACTLR_EL1 and AArch64-IMP_CLUSTERACTLR_EL1 at EL1 to be trapped. **0b1** This control does not cause any instructions to be trapped.	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

Accessibility

MRS <Xt>, ACTLR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return ACTLR_EL2;
elsif PSTATE.EL == EL3 then
    return ACTLR_EL2;
```

MSR ACTLR_EL2, <Xt>

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

```
    ACTLR_EL2 = X[t];
```

A.1.17 HACR_EL2, Hypervisor Auxiliary Control Register

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

Note

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

Configurations

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Figure A-18: AArch64_hacr_el2 bit assignments

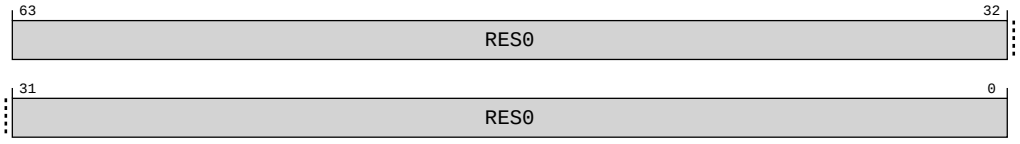


Table A-57: HACR_EL2 bit descriptions

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

MSR HACR_EL2, <Xt>

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

A.1.18 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

See bit descriptions

Reset value

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

Figure A-19: AArch64_afsr0_el2 bit assignments

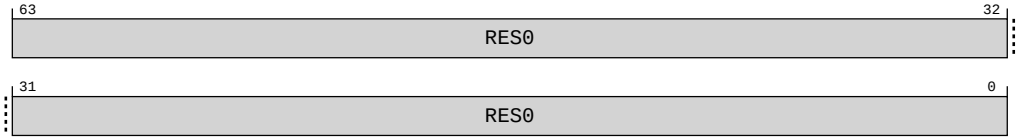


Table A-60: AFSR0_EL2 bit descriptions

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

Access

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic AFSR0_EL2 or AFSR0_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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 AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic AFSR0_EL2 or AFSR0_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

MRS <Xt>, AFSR0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return AFSR0_EL2;
elsif PSTATE.EL == EL3 then
    return AFSR0_EL2;
```

MSR AFSR0_EL2, <Xt>

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

MRS <Xt>, AFSR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR0_EL2;
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    return AFSR0_EL1;
```

MSR AFSR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL2 = X[t];
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t];
```

A.1.19 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

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-20: AArch64_afsr1_el2 bit assignments

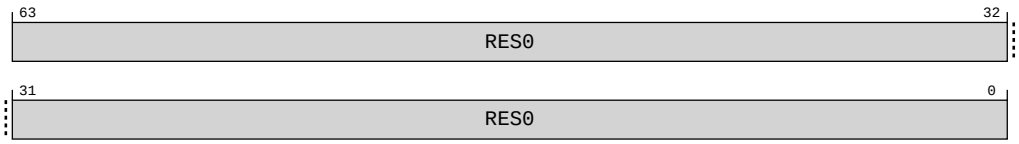


Table A-65: AFSR1_EL2 bit descriptions

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

Access

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic AFSR1_EL2 or AFSR1_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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 AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic AFSR1_EL2 or AFSR1_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.
MRS <Xt>, AFSR1_EL2

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

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

MSR AFSR1_EL2, <Xt>

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

MRS <Xt>, AFSR1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR1_EL2;
    else
        return AFSR1_EL1;
elsif PSTATE.EL == EL3 then
    return AFSR1_EL1;
```

MSR AFSR1_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AFSR1_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL2 = X[t];
    else
        AFSR1_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR1_EL1 = X[t];
```

A.1.20 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by AArch64-MAIR_EL2.

Configurations

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

AMAIR_EL2 is permitted to be cached in a TLB.

Figure A-21: AArch64_amair_el2 bit assignments

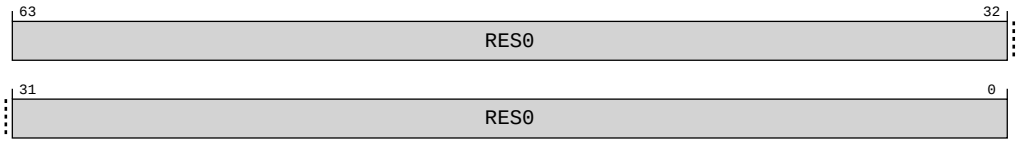


Table A-70: AMAIR_EL2 bit descriptions

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

Access

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic AMAIR_EL2 or AMAIR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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 AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic AMAIR_EL2 or AMAIR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.
MRS <Xt>, AMAIR_EL2

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

MSR AMAIR_EL2, <Xt>

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

MRS <Xt>, AMAIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AMAIR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AMAIR_EL2;
    else
        return AMAIR_EL1;
elsif PSTATE.EL == EL3 then
    return AMAIR_EL1;
```

MSR AMAIR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AMAIR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL2 = X[t];
    else
        AMAIR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AMAIR_EL1 = X[t];
```

A.1.21 IMP_ATCR_EL2, CPU Auxiliary Translation Control Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-22: AArch64_imp_atcr_el2 bit assignments

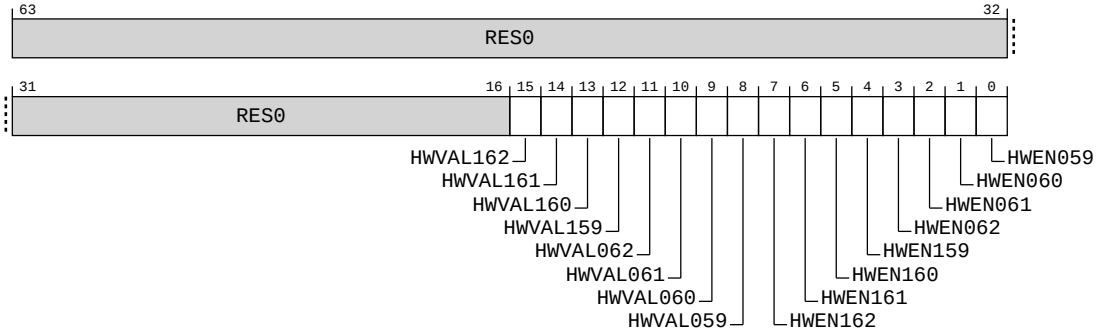


Table A-75: IMP_ATCR_EL2 bit descriptions

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

Bits	Name	Description	Reset
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	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 EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return IMP_ATCR_EL2;
elsif PSTATE.EL == EL3 then
    return IMP_ATCR_EL2;
```

MSR S3_4_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_ATCR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL2 = X[t];
```

A.1.22 IMP_AVTCR_EL2, CPU Auxiliary Translation Control Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-23: AArch64_imp_avtcr_el2 bit assignments

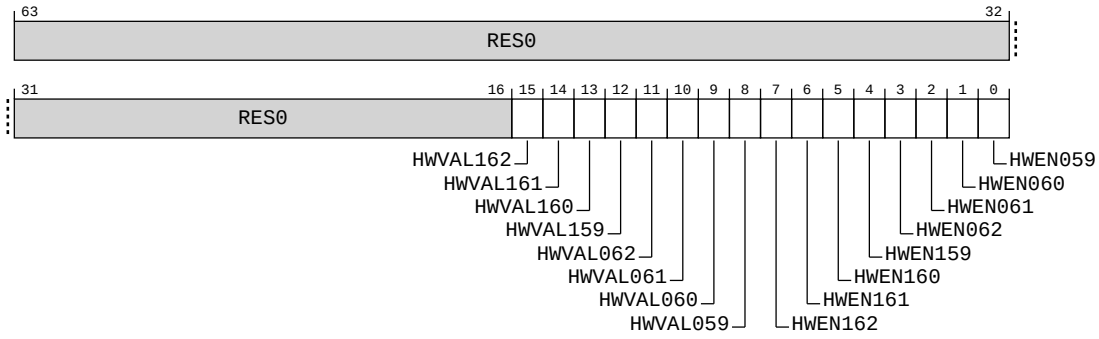


Table A-78: IMP_AVTCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN162 is set.	x
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN161 is set.	x

Bits	Name	Description	Reset
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN160 is set.	x
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN159 is set.	x
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using VTTBR_EL2 if HWEN062 is set.	x
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using VTTBR_EL2 if HWEN061 is set.	x
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using VTTBR_EL2 if HWEN060 is set.	x
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using VTTBR_EL2 if HWEN059 is set.	x
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	0b0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	0b0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	0b0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	0b0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using VTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	0b0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using VTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using VTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using VTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	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 EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return IMP_AVTCR_EL2;
elsif PSTATE.EL == EL3 then
    return IMP_AVTCR_EL2;
```

MSR S3_4_C15_C7_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_AVTCR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_AVTCR_EL2 = X[t];
```

A.1.23 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

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 0xxx 0xxx xx00

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-24: AArch64_actlr_el3 bit assignments

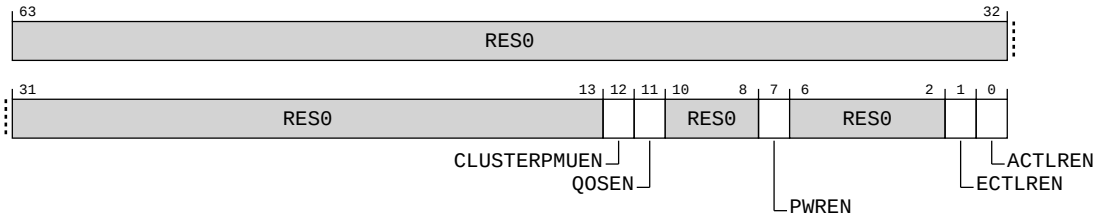


Table A-81: ACTLR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:13]	**RES0**	Reserved	**RES0**
[12]	CLUSTERPMUEN	Cluster PMU Registers enable. Traps EL1 and EL2 writes to implementation-defined cluster PMU registers to EL3, subject to the exception prioritization rules. Possible values of this bit are: **0b0** This control causes writes to IMP_CLUSTERPM* at EL1 and EL2 to be trapped to EL3, subject to the exception prioritization rules.. **0b1** This control does not cause any instructions to be trapped.	0b0
[11]	QOSEN	Cluster Bus QoS Registers enable. Traps EL1 and EL2 writes to AArch64-IMP_CLUSTERBUSQOS_EL1 to EL3, subject to the exception prioritization rules. Possible values of this bit are: **0b0** This control causes writes to AArch64-IMP_CLUSTERBUSQOS_EL1 at EL1 and EL2 to be trapped to EL3, subject to the exception prioritization rules. **0b1** This control does not cause any instructions to be trapped.	0b0
[10:8]	**RES0**	Reserved	**RES0**
[7]	PWREN	Power Control Registers enable. Traps EL1 and EL2 writes to implementation-defined power control registers to EL3, subject to the exception prioritization rules. Possible values of this bit are: **0b0** This control causes writes to AArch64-IMP_CPUPWRCTLR_EL1, AArch64-IMP_CLUSTERPWRCTLR_EL1, AArch64-IMP_CLUSTERPWRDN_EL1 and IMP_CLUSTERL3*_EL1 at EL1 and EL2 to be trapped to EL3, subject to the exception prioritization rules. **0b1** This control does not cause any instructions to be trapped.	0b0
[6:2]	**RES0**	Reserved	**RES0**
[1]	ECTLREN	Extended Control Registers enable. Traps EL1 and EL2 writes to AArch64-IMP_CPUECTLR_EL1, AArch64-IMP_CMPXECTLR_EL1 and AArch64-IMP_CLUSTERECTLR_EL1 to EL3, subject to the exception prioritization rules. Possible values of this bit are: **0b0** This control causes writes to AArch64-IMP_CPUECTLR_EL1, AArch64-IMP_CMPXECTLR_EL1 and AArch64-IMP_CLUSTERECTLR_EL1 at EL1 and EL2 to be trapped to EL3, subject to the exception prioritization rules. **0b1** This control does not cause any instructions to be trapped.	0b0
[0]	ACTLREN	Auxiliary Control Registers enable. Traps EL1 and EL2 writes to AArch64-IMP_CPUACTLR_EL1, AArch64-IMP_CPUACTLR2_EL1, AArch64-IMP_CMPXACTLR_EL1 and AArch64-IMP_CLUSTERACTLR_EL1 to EL3, subject to the exception prioritization rules. Possible values of this bit are: **0b0** This control causes writes to AArch64-IMP_CPUACTLR_EL1, AArch64-IMP_CPUACTLR2_EL1, AArch64-IMP_CMPXACTLR_EL1 and AArch64-IMP_CLUSTERACTLR_EL1 at EL1 and EL2 to be trapped to EL3, subject to the exception prioritization rules. **0b1** This control does not cause any instructions to be trapped.	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
    return ACTLR_EL3;
```

MSR ACTLR_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    ACTLR_EL3 = X[t];
```

A.1.24 AFSR0_EL3, Auxiliary Fault Status Register 0 (EL3)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Figure A-25: AArch64_afsr0_el3 bit assignments

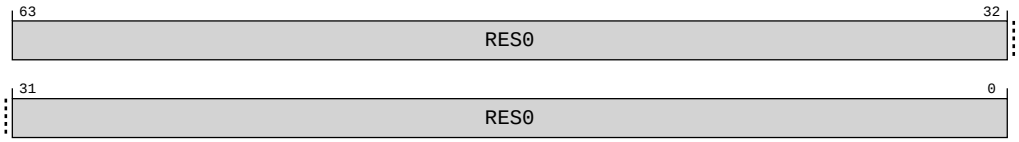


Table A-84: AFSR0_EL3 bit descriptions

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
    return AFSR0_EL3;
```

MSR AFSR0_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AFSR0_EL3 = X[t];
```

A.1.25 AFSR1_EL3, Auxiliary Fault Status Register 1 (EL3)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Figure A-26: AArch64_afsr1_el3 bit assignments

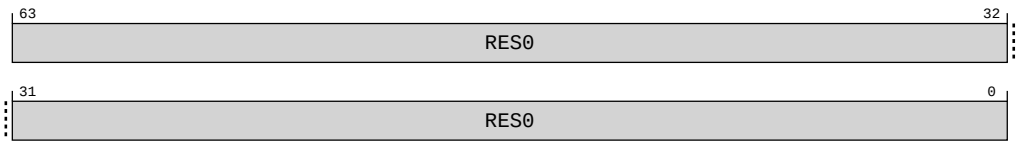


Table A-87: AFSR1_EL3 bit descriptions

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
    return AFSR1_EL3;
```

MSR AFSR1_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AFSR1_EL3 = X[t];
```

A.1.26 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register (EL3)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by AArch64-MAIR_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

AMAIR_EL3 is permitted to be cached in a TLB.

Figure A-27: AArch64_amair_el3 bit assignments

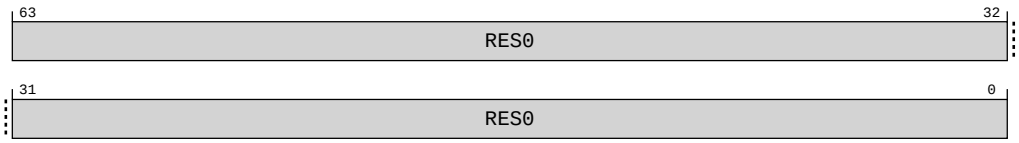


Table A-90: AMAIR_EL3 bit descriptions

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
    return AMAIR_EL3;
```

MSR AMAIR_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AMAIR_EL3 = X[t];
```

A.1.27 IMP_ATCR_EL3, CPU Auxiliary Translation Control Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-28: AArch64_imp_atcr_el3 bit assignments

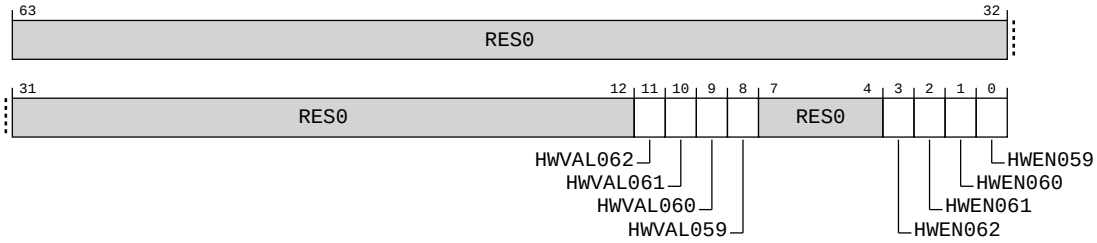


Table A-93: IMP_ATCR_EL3 bit descriptions

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

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

MSR S3_6_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL3 = X[t];
```

A.2 AArch64 Special-purpose registers summary

The summary table provides an overview of all Special-purpose 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-96: Special-purpose registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SPSR_EL1	3	0	C4	C0	0	–	64-bit	Saved Program Status Register (EL1)
ELR_EL1	3	0	C4	C0	1	–	64-bit	Exception Link Register (EL1)
SP_EL0	3	0	C4	C1	0	–	64-bit	Stack Pointer (EL0)
DSPSR_EL0	3	3	C4	C5	0	–	64-bit	Debug Saved Program Status Register
DLR_EL0	3	3	C4	C5	1	–	64-bit	Debug Link Register
SPSR_EL2	3	4	C4	C0	0	–	64-bit	Saved Program Status Register (EL2)
ELR_EL2	3	4	C4	C0	1	–	64-bit	Exception Link Register (EL2)
SP_EL1	3	4	C4	C1	0	–	64-bit	Stack Pointer (EL1)
SPSR_irq	3	4	C4	C3	0	–	64-bit	Saved Program Status Register (IRQ mode)
SPSR_abt	3	4	C4	C3	1	–	64-bit	Saved Program Status Register (Abort mode)
SPSR_und	3	4	C4	C3	2	–	64-bit	Saved Program Status Register (Undefined mode)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SPSR_fiq	3	4	C4	C3	3	–	64-bit	Saved Program Status Register (FIQ mode)
SPSR_EL3	3	6	C4	C0	0	–	64-bit	Saved Program Status Register (EL3)
ELR_EL3	3	6	C4	C0	1	–	64-bit	Exception Link Register (EL3)
SP_EL2	3	6	C4	C1	0	–	64-bit	Stack Pointer (EL2)
IMP_CPUPPMCR_EL3	3	6	C15	C2	0	–	64-bit	Global PPM Configuration Register

A.2.1 IMP_CPUPPMCR_EL3, Global PPM Configuration Register

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

Configurations

AArch64 register IMP_CPUPPMCR_EL3 bits [63:0] are architecturally mapped to External System register B.1.1 CPUPPMCR, Global PPM Configuration Register on page 460 bits [63:0].

Attributes

Width

64

Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-29: AArch64_imp_cpuppmcr_el3 bit assignments

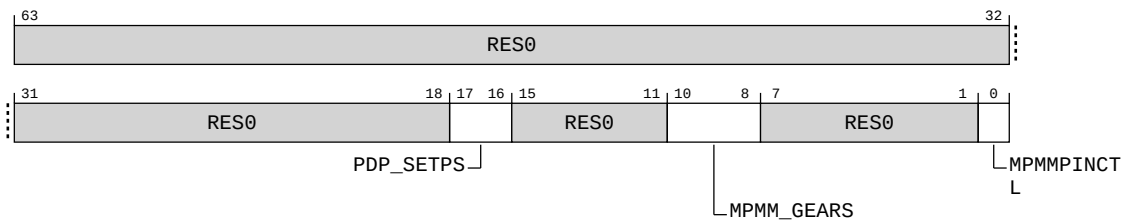


Table A-97: IMP_CPUPPMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:18]	**RES0**	Reserved	**RES0**
[17:16]	PDP_SETPS	Number of PDP Setpoints implemented **0b00** PDP is not implemented or enabled. Access to this field is: RO	xx
[15:11]	**RES0**	Reserved	**RES0**
[10:8]	MPMM_GEARS	Number of MPMM Gears implemented **0b011** 3 MPMM are enabled. Access to this field is: RO	xxx
[7:1]	**RES0**	Reserved	**RES0**
[0]	MPMMPINCTL	MPMM Pin Control Enabled **0b0** MPMM control through SPR and utility bus. **0b1** MPMM control through pin only.	0b0

Access

MRS <Xt>, S3_6_C15_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
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR_EL3;
```

MSR S3_6_C15_C2_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR_EL3 = X[t];
```

A.3 AArch64 Debug registers summary

The 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-100: Debug registers summary

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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
DBGBCR3_EL1	2	0	C0	C3	5	–	64-bit	Debug Breakpoint Control Registers
DBGWVR3_EL1	2	0	C0	C3	6	–	64-bit	Debug Watchpoint Value Registers
DBGWCR3_EL1	2	0	C0	C3	7	–	64-bit	Debug Watchpoint Control Registers
DBGBVR4_EL1	2	0	C0	C4	4	–	64-bit	Debug Breakpoint Value Registers
DBGBCR4_EL1	2	0	C0	C4	5	–	64-bit	Debug Breakpoint Control Registers
DBGBVR5_EL1	2	0	C0	C5	4	–	64-bit	Debug Breakpoint Value Registers
DBGBCR5_EL1	2	0	C0	C5	5	–	64-bit	Debug Breakpoint Control Registers
OSECCR_EL1	2	0	C0	C6	2	–	64-bit	OS Lock Exception Catch Control Register
MDRAR_EL1	2	0	C1	C0	0	–	64-bit	Monitor Debug ROM Address Register
OSLAR_EL1	2	0	C1	C0	4	–	64-bit	OS Lock Access Register
OSLSR_EL1	2	0	C1	C1	4	–	64-bit	OS Lock Status Register
OSDLR_EL1	2	0	C1	C3	4	–	64-bit	OS Double Lock Register
DBGPRCR_EL1	2	0	C1	C4	4	–	64-bit	Debug Power Control Register
DBGCLAIMSET_EL1	2	0	C7	C8	6	–	64-bit	Debug CLAIM Tag Set register
DBGCLAIMCLR_EL1	2	0	C7	C9	6	–	64-bit	Debug CLAIM Tag Clear register
DBGAUTHSTATUS_EL1	2	0	C7	C14	6	–	64-bit	Debug Authentication Status register
MDCCSR_EL0	2	3	C0	C1	0	–	64-bit	Monitor DCC Status Register
DBGDTR_EL0	2	3	C0	C4	0	–	64-bit	Debug Data Transfer Register, half-duplex
DBGDTRRX_EL0	2	3	C0	C5	0	–	64-bit	Debug Data Transfer Register, Receive
DBGDTRTX_EL0	2	3	C0	C5	0	–	64-bit	Debug Data Transfer Register, Transmit
TRFCR_EL1	3	0	C1	C2	1	–	64-bit	Trace Filter Control Register (EL1)
MDCR_EL2	3	4	C1	C1	1	–	64-bit	Monitor Debug Configuration Register (EL2)
TRFCR_EL2	3	4	C1	C2	1	–	64-bit	Trace Filter Control Register (EL2)
IMP_CDBGDR0_EL3	3	6	C15	C0	0	–	64-bit	Cache Debug Data Register 0

A.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0

Contains data from a preceding cache debug operation.

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

- SYS IMP_CDBGL1DCDR
- SYS IMP_CDBGL1DCMR
- SYS IMP_CDBGL1DCTR
- SYS IMP_CDBGL1ICDR
- SYS IMP_CDBGL1ICTR
- SYS IMP_CDBGL2CDR
- SYS IMP_CDBGL2CMR
- SYS IMP_CDBGL2CTR

- SYS IMP_CDBGL2TR0
- SYS IMP_CDBGL2TR1
- SYS IMP_CDBGL2TR2

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

When SYS IMP_CDBGL1DCDR or SYS IMP_CDBGL2CDR is executed

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

When SYS IMP_CDBGL1DCMR or SYS IMP_CDBGL2CMR is executed

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

When SYS IMP_CDBGL1ICDR is executed

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

When SYS IMP_CDBGL1DCTR is executed

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

When SYS IMP_CDBGL1DCDTR is executed

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

When SYS IMP_CDBGL1ICTR is executed

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

When SYS IMP_CDBGL2CTR is executed

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

When SYS IMP_CDBGL2TR0 is executed

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

When SYS IMP_CDBGL2TR1 is executed

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

When SYS IMP_CDBGL2TR2 is executed

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 SYS IMP_CDBGL1DCDR or SYS IMP_CDBGL2CDR is executed

Figure A-30: AArch64_imp_cdbgdr0_el3 bit assignments

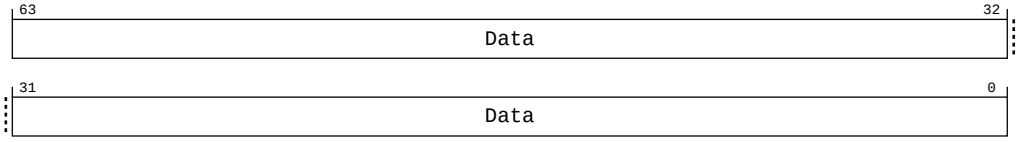


Table A-101: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Data	Data contents of cache at specified Set/Way/Offset	64{x}

When SYS IMP_CDBGL1DCMR or SYS IMP_CDBGL2CMR is executed

Figure A-31: AArch64_imp_cdbgdr0_el3 bit assignments

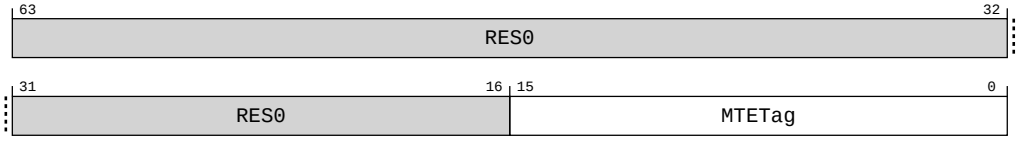


Table A-102: IMP_CDBGDR0_EL3 bit descriptions

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

When SYS IMP_CDBGL1ICDR is executed

Figure A-32: AArch64_imp_cdbgdr0_el3 bit assignments

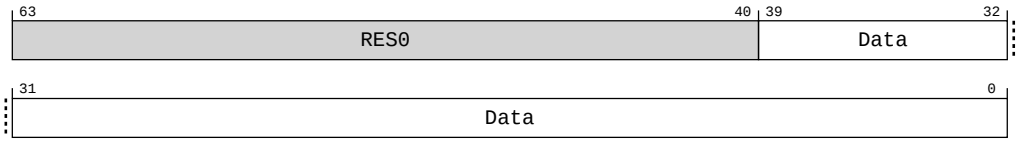


Table A-103: IMP_CDBGDR0_EL3 bit descriptions

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

When SYS IMP_CDBGL1DCTR is executed

Figure A-33: AArch64_imp_cdbgdr0_el3 bit assignments

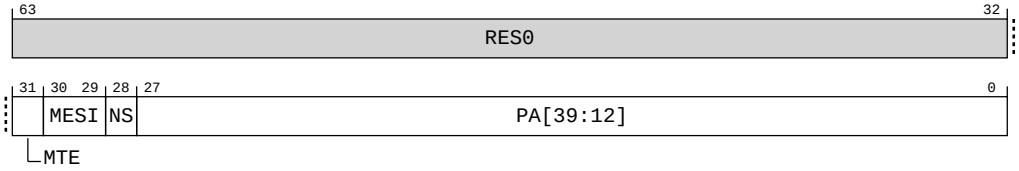


Table A-104: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	MTE	MTE tag state **0b0** MTE tag invalid **0b1** MTE tag valid	x
[30:29]	MESI	Partial MESI state **0b00** Invalid **0b01** Shared **0b10** Unique Non-transient **0b11** Unique Transient	xx
[28]	NS	Tag security state **0b0** Secure **0b1** Non-secure	x
[27:0]	PA[39:12]	Tag physical address	28{x}

When SYS IMP_CDBGL1DCDTR is executed

Figure A-34: AArch64_imp_cdbgdr0_el3 bit assignments

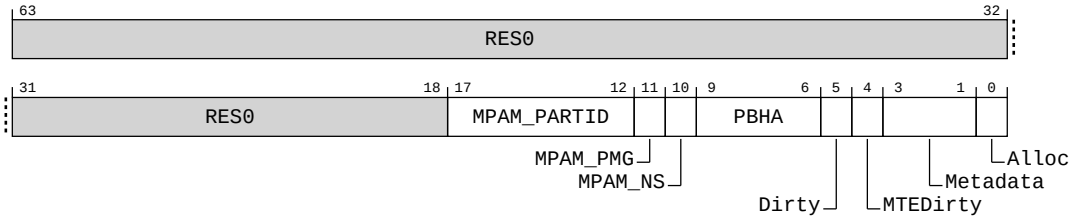


Table A-105: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:18]	**RES0**	Reserved	**RES0**
[17:12]	MPAM_PARTID	MPAM partition ID	6{x}
[11]	MPAM_PMG	MPAM performance monitoring group	x
[10]	MPAM_NS	Indicates MPAM PARTID space **0b0** Secure physical PARTID space **0b1** Non-secure physical PARTID space	x
[9:6]	PBHA	Page-Based Hardware Attributes	xxxx
[5]	Dirty	Indicates whether the cache line data is dirty	x
[4]	MTEDirty	Indicates whether the MTE tag data for the cache line is dirty	x
[3:1]	Metadata	Internal metadata	xxx
[0]	Alloc	Outer allocation hint **0b0** No write allocate **0b1** Write allocate	x

When SYS IMP_CDBGL1ICTR is executed

Figure A-35: AArch64_imp_cdbgdr0_el3 bit assignments

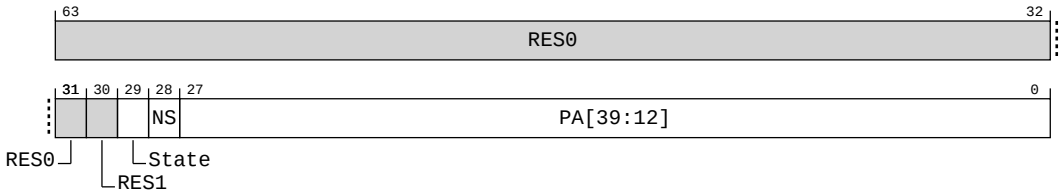


Table A-106: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**

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

When SYS IMP_CDBGL2CTR is executed

Figure A-36: AArch64_imp_cdbgdr0_el3 bit assignments

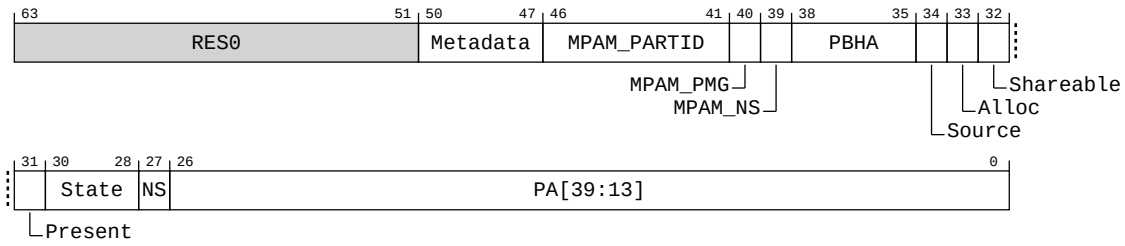


Table A-107: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:51]	**RES0**	Reserved	**RES0**
[50:47]	Metadata	Internal metadata	xxxx
[46:41]	MPAM_PARTID	MPAM partition ID	6{x}
[40]	MPAM_PMG	MPAM performance monitoring group	x
[39]	MPAM_NS	Indicates MPAM PARTID space **0b0** Secure physical PARTID space **0b1** Non-secure physical PARTID space	x
[38:35]	PBHA	Page-Based Hardware Attributes	xxxx
[34]	Source	Cache line source **0b0** Line was brought into complex from outside the cluster **0b1** Line was brought into complex from an L3 hit	x

Bits	Name	Description	Reset
[33]	Alloc	Outer allocation hint **0b0** No write allocate **0b1** Write allocate	x
[32]	Shareable	Cache line shareability **0b0** Non-shareable **0b1** Outer shareable	x
[31]	Present	Cache line is present in the L1 cache of any of the cores in this complex.	x
[30:28]	State	Cache line state **0b000** Invalid. Line can be considered Invalid also when bit [50] is 0b1. **0b001** SharedClean, MTE tags invalid **0b010** UniqueClean, MTE tags invalid **0b011** UniqueDirty, MTE tags invalid **0b100** SharedClean, MTE tags clean **0b101** UniqueClean, MTE tags clean **0b110** UniqueDirty, MTE tags clean **0b111** UniqueDirty, MTE tags dirty	xxx
[27]	NS	Tag security state **0b0** Secure **0b1** Non-secure	x
[26:0]	PA[39:13]	Tag physical address. Depending on L2 cache size, bits [2:0] might be **RES0**.	27{x}

When SYS IMP_CDBGL2TR0 is executed

Figure A-37: AArch64_imp_cdbgdr0_el3 bit assignments

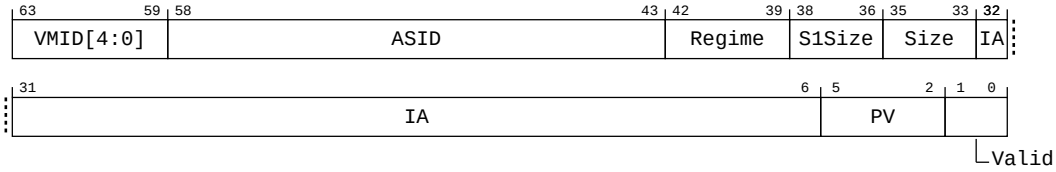


Table A-108: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:59]	VMID[4:0]	Lower bits of the VMID value, when supported by the regime	5{x}
[58:43]	ASID	ASID value, when supported by the regime	16{x}
[42:39]	Regime	Translation regime used to fetch the entry **0b0000** Secure EL1&0 **0b0001** Secure EL2&0 **0b0010** Secure EL2 **0b0011** Secure EL3 **0b0100** Non-secure EL1&0 **0b0101** Non-secure EL2&0 **0b0110** Non-secure EL2 **1xxx** RESERVED	xxxx

Bits	Name	Description	Reset
[38:36]	S1Size	The original size of the stage 1 translation **0b000** 4KB or 16KB **0b001** 64KB **0b010** 2MB **0b011** 8MB **0b100** 32MB **0b101** 128MB **0b110** 512MB **0b111** 1GB	xxx
[35:33]	Size	The size of the entry **0b000** 16KB **0b001** 64KB **0b010** 2MB **0b011** 8MB **0b100** 32MB **0b101** 128MB **0b110** 512MB **0b111** 1GB	xxx

Bits	Name	Description	Reset
32:6	IA	**IA encoding for For main TLB entries and walk entries** **26:0** Input virtual address of the entry **IA encoding for For IPA entries** **26** NS bit of input intermediate physical address of the entry **25:7** Input intermediate physical address of the entry **6:0** Reserved, **RES0**.	27{x}
5:2	PV	**IA encoding for For main TLB entries and walk entries** **26:0** Input virtual address of the entry **IA encoding for For IPA entries** **26** NS bit of input intermediate physical address of the entry **25:7** Input intermediate physical address of the entry **6:0** Reserved, **RES0**. **PV encoding for For main TLB entries** **3:0** For 16KB entries indicates if individual 4KB mappings are valid. **PV encoding for For medium and large entries** **3** For walk entries, specifies if the stage 1 walk is secure or non-secure. **2** For IPA entries, specifies whether the mapping size was influenced by the contiguous hint. **1:0** Indicates type of entry. **0b01** Walk entry **0b10** IPA entry	xxxx

Bits	Name	Description	Reset
[1:0]	Valid	TLB entry valid (one bit per core). When both bits are set the entry is CnP. **0b00** Invalid **0b01** Valid, core 0 private **0b10** Valid, core 1 private **0b11** Valid, common	xx

When SYS IMP_CDBGL2TR1 is executed

Figure A-38: AArch64_imp_cdbgdr0_el3 bit assignments

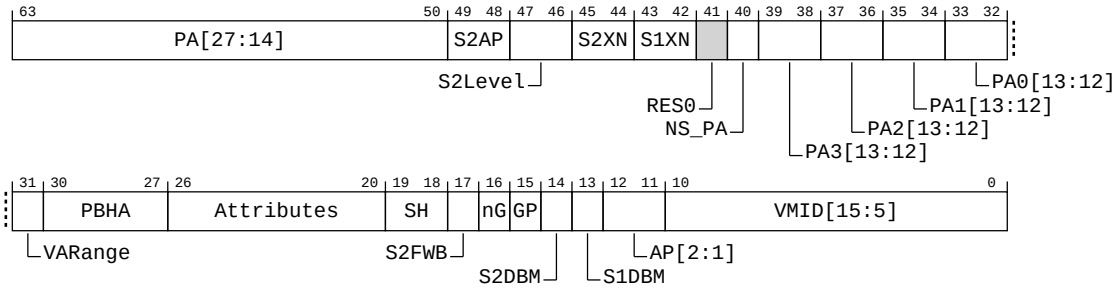


Table A-109: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:50]	PA[27:14]	The lower bits [14:13] of the PA	14{x}
[49:48]	S2AP	Stage 2 access permissions	xx
[47:46]	S2Level	Final level of stage 2 page walk used to generate PA	xx
[45:44]	S2XN	S2 execute never permissions	xx
[43:42]	S1XN	S1 execute never permissions	xx
[41]	**RES0**	Reserved	**RES0**
[40]	NS_PA	NS bit for the PA space	x
[39:38]	PA3[13:12]	Low bits of PA for the clustered entry with VA[13:12] == 3 (only for 16k pages)	xx
[37:36]	PA2[13:12]	Low bits of PA for the clustered entry with VA[13:12] == 2 (only for 16k pages)	xx
[35:34]	PA1[13:12]	Low bits of PA for the clustered entry with VA[13:12] == 1 (only for 16k pages)	xx
[33:32]	PA0[13:12]	Low bits of PA for the clustered entry with VA[13:12] == 0 (only for 16k pages)	xx
[31]	VARange	The VA range for translation regimes which support two VA ranges **0b0** Lower VA range **0b1** Upper VA range	x
[30:27]	PBHA	Page-Based Hardware Attributes	xxxx

Bits	Name	Description	Reset
[26:20]	Attributes	Memory attributes for the entry **x000x00** Device-nGnRnE. **x000x01** Device-nGnRE. **x000x10** Device-nGRE. **x000x11** Device-GRE. **x0010:Outer[1:0]** Normal memory, Inner Non-cacheable, Outer Write-Back. Outer[1:0] are the outer allocation hints. **x0011:Outer[1:0]** Normal memory, Inner Write-Through, Outer Write-Back. Outer[1:0] are the outer allocation hints. **x0100:Outer[1:0]** Normal memory, Inner Non-cacheable, Outer Write-Through. Outer[1:0] are the outer allocation hints. **x0101:Outer[1:0]** Normal memory, Inner Write-Back, Outer Write-Through. Outer[1:0] are the outer allocation hints. **x0110:Outer[1:0]** Normal memory, Inner Write-Through, Outer Write-Through. Outer[1:0] are the outer allocation hints. **x011100** Normal memory, Inner Non-cacheable, Outer Non-cacheable. **x011101** Normal memory, Inner Write-Back, Outer Non-cacheable. **x011110** Normal memory, Inner Write-Through, Outer Non-cacheable. **010:Inner[1:0]:Outer[1:0]** Normal memory, Inner Write-Back, Outer Write-Back Non-transient. Inner[1:0] are the inner allocation hints and Outer[1:0] are the outer allocation hints. **011:Inner[1:0]:Outer[1:0]** Normal memory, Inner Write-Back, Outer Write-Back Transient. Inner[1:0] are the inner allocation hints and Outer[1:0] are the outer allocation hints. **110:Inner[1:0]:Outer[1:0]** Tagged Normal memory, Inner Write-Back, Outer Write-Back Non-transient. Inner[1:0] are the inner allocation hints and Outer[1:0] are the outer allocation hints.	7{x}

Bits	Name	Description	Reset
[19:18]	SH	Shareability **0b00** Non-shareable **0b10** Outer shareable **0b11** Inner shareable **Note:** Device memory is always outer shareable.	xx
[17]	S2FWB	Stage 2 forced attributes to be WB	x
[16]	nG	Not global	x
[15]	GP	Guarded page	x
[14]	S2DBM	Stage 2 Dirty Bit Modifier	x
[13]	S1DBM	Stage 1 Dirty Bit Modifier	x
[12:11]	AP[2:1]	Stage 1 access permissions	xx
[10:0]	VMID[15:5]	Upper bits of the VMID value, when supported by the regime	11{x}

When SYS IMP_CDBGL2TR2 is executed

Figure A-39: AArch64_imp_cdbgdr0_el3 bit assignments

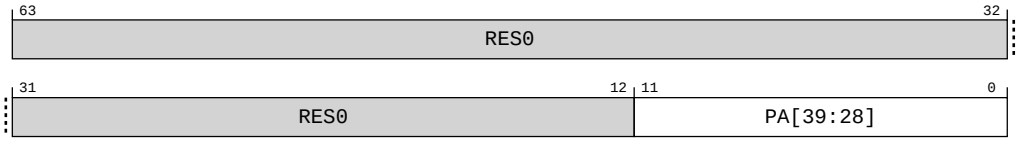


Table A-110: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:12]	**RES0**	Reserved	**RES0**
[11:0]	PA[39:28]	The Upper bits [39:28] of the PA	12{x}

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
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CDBGDR0_EL3;
```

A.4 AArch64 Cache Debug instructions summary

The summary table provides an overview of all Cache Debug instructions 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-112: Cache Debug instructions summary

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

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

Read contents of the L1 Data Cache Tag Memory.

The cache tag is written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-40: AArch64_sys_imp_cdbgl1dctr bit assignments

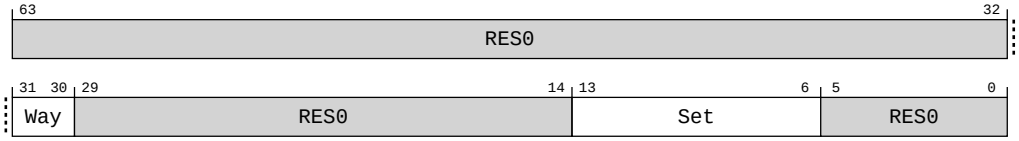


Table A-113: SYS IMP_CDBGL1DCTR bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:30]	Way	Cache way	xx
[29:14]	**RES0**	Reserved	**RES0**
[13:6]	Set	Cache set	8{x}
[5:0]	**RES0**	Reserved	**RES0**

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

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

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0010	0b000

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.
SYS #6, C15, C2, #0{, <Xt>}

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

```
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1DCTR(X[t]);
```

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

Read contents of the L1 Instruction Cache Tag Memory.

The cache tag is written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-41: AArch64_sys_imp_cdbgl1ictr bit assignments

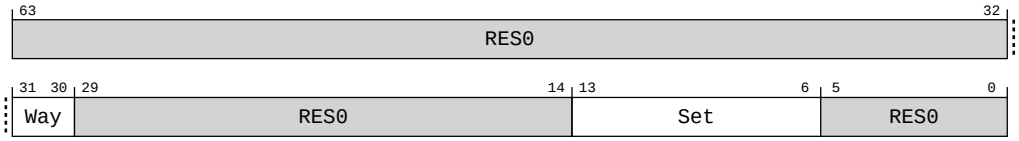


Table A-115: SYS IMP_CDBGL1ICTR bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:30]	Way	Cache way	xx
[29:14]	**RES0**	Reserved	**RES0**
[13:6]	Set	Cache set	8{x}
[5:0]	**RES0**	Reserved	**RES0**

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

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

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0010	0b001

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.
SYS #6, C15, C2, #1{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1ICTR(X[t]);
```

A.4.3 SYS IMP_CDBGL2TR0, L2 TLB Read Operation 0

Read contents of the Level 2 TLB Memory.

Bits [63:0] of the TLB data are written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-42: AArch64_sys_imp_cdbgl2tr0 bit assignments

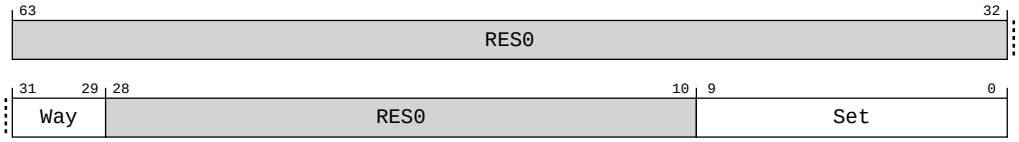


Table A-117: SYS IMP_CDBGL2TR0 bit descriptions

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

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary TLB entry.

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

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0010	0b010

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary TLB entry.
SYS #6, C15, C2, #2{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2TR0(X[t]);
```

A.4.4 SYS IMP_CDBGL2CTR, L2 Cache Tag Read Operation

Read contents of the L2 Cache Tag Memory.

The cache tag is written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-43: AArch64_sys_imp_cdbgl2ctr bit assignments

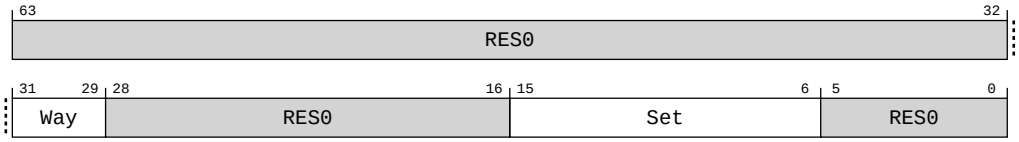


Table A-119: SYS IMP_CDBGL2CTR bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:29]	Way	Cache way	xxx
[28:16]	**RES0**	Reserved	**RES0**
[15:6]	Set	Cache set	10{x}
[5:0]	**RES0**	Reserved	**RES0**

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

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

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0010	0b011

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.
SYS #6, C15, C2, #3{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2CTR(X[t]);
```

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

Read contents of the L1 Data Cache Dirty Memory.

The cache tag is written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-44: AArch64_sys_imp_cdbgl1dcdtr bit assignments

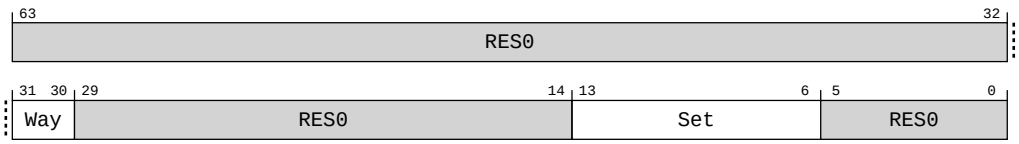


Table A-121: SYS IMP_CDBGL1DCDTR bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[31:30]	Way	Cache way	xx
[29:14]	**RES0**	Reserved	**RES0**
[13:6]	Set	Cache set	8{x}
[5:0]	**RES0**	Reserved	**RES0**

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

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

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0010	0b100

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.
SYS #6, C15, C2, #4{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1DCDTR(X[t]);
```

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

Read contents of the L1 Data Cache MTE Tag Memory.

The 16 bits of cache MTE tag data are written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-45: AArch64_sys_imp_cdbgl1dcmr bit assignments

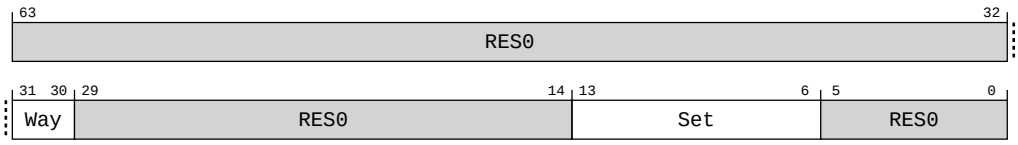


Table A-123: SYS IMP_CDBGL1DCMR bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:30]	Way	Cache way	xx
[29:14]	**RES0**	Reserved	**RES0**
[13:6]	Set	Cache set	8{x}
[5:0]	**RES0**	Reserved	**RES0**

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

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

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0011	0b000

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.
SYS #6, C15, C3, #0{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1DCMR(X[t]);
```

A.4.7 SYS IMP_CDBGL2TR1, L2 TLB Read Operation 1

Read contents of the Level 2 TLB Memory.

Bits [127:64] of the TLB data are written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-46: AArch64_sys_imp_cdbgl2tr1 bit assignments

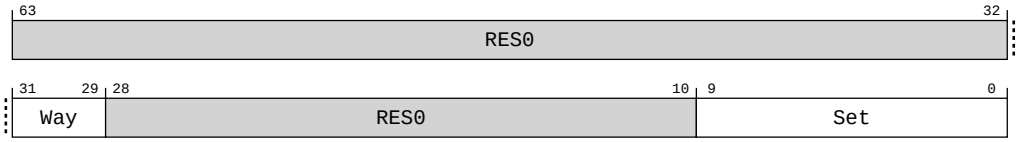


Table A-125: SYS IMP_CDBGL2TR1 bit descriptions

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

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary TLB entry.

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

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0011	0b010

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary TLB entry.
SYS #6, C15, C3, #2{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2TR1(X[t]);
```

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

Read contents of the L2 Cache MTE Tag Memory.

The 16 bits of cache MTE tag data are written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-47: AArch64_sys_imp_cdbgl2cmr bit assignments

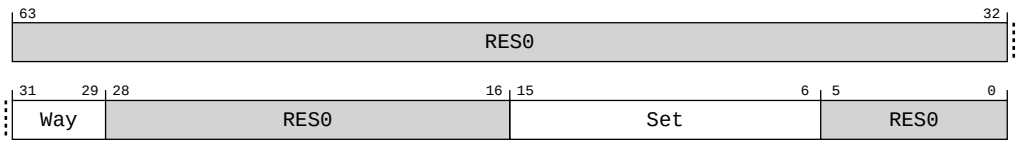


Table A-127: SYS IMP_CDBGL2CMR bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[31:29]	Way	Cache way	xxx
[28:16]	RES0	Reserved	RES0
[15:6]	Set	Cache set	10{x}
[5:0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

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

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0011	0b011

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.
SYS #6, C15, C3, #3{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2CMR(X[t]);
```

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

Read contents of the L1 Data Cache Data Memory.

The 64 bits of cache data are written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-48: AArch64_sys_imp_cdbgl1dcdr bit assignments

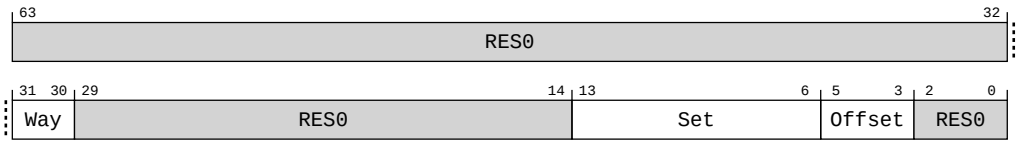


Table A-129: SYS IMP_CDBGL1DCDR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:30]	Way	Cache way	xx
[29:14]	RES0	Reserved	RES0
[13:6]	Set	Cache set	8{x}
[5:3]	Offset	Cache data element offset	xxx
[2:0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

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

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0100	0b000

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.
SYS #6, C15, C4, #0{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1DCDR(X[t]);
```

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

Read contents of the L1 Instruction Cache Data Memory.

The 40 bits of cache data are written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-49: AArch64_sys_imp_cdbgl1icdr bit assignments

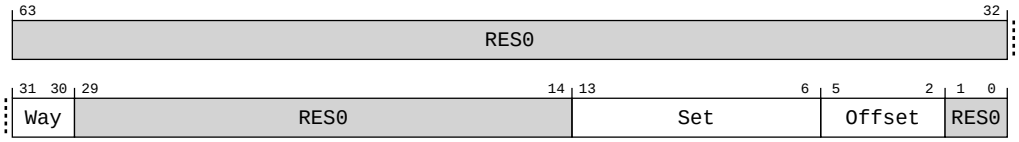


Table A-131: SYS IMP_CDBGL1ICDR bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:30]	Way	Cache way	xx
[29:14]	**RES0**	Reserved	**RES0**
[13:6]	Set	Cache set	8{x}
[5:2]	Offset	Cache data element offset	xxxx
[1:0]	**RES0**	Reserved	**RES0**

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

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

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0100	0b001

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.
SYS #6, C15, C4, #1{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1ICDR(X[t]);
```

A.4.11 SYS IMP_CDBGL2TR2, L2 TLB Read Operation 2

Read contents of the Level 2 TLB Memory.

Bits [191:128] of the TLB data are written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-50: AArch64_sys_imp_cdbgl2tr2 bit assignments

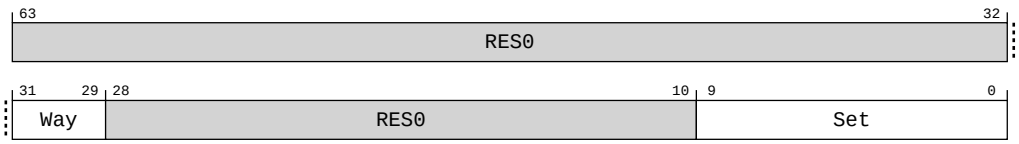


Table A-133: SYS IMP_CDBGL2TR2 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**

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

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary TLB entry.

SYS #6, C15, C4, #2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0100	0b010

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary TLB entry.
SYS #6, C15, C4, #2{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2TR2(X[t]);
```

A.4.12 SYS IMP_CDBGL2CDR, L2 Cache Data Read Operation

Read contents of the L2 Cache Data Memory.

The 64 bits of cache data are written to AArch64-IMP_CDBGDR0_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Cache Debug instructions

Access type

See bit descriptions

Bit descriptions

Figure A-51: AArch64_sys_imp_cdbgl2cdr bit assignments

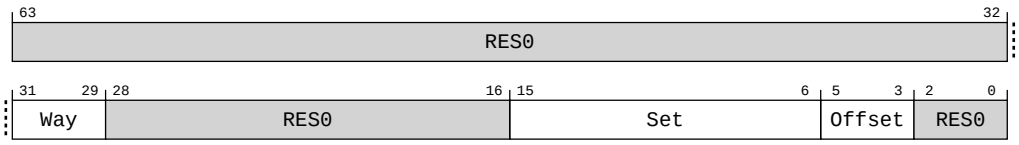


Table A-135: SYS IMP_CDBGL2CDR bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:29]	Way	Cache way	xxx
[28:16]	**RES0**	Reserved	**RES0**
[15:6]	Set	Cache set	10{x}
[5:3]	Offset	Cache data element offset	xxx
[2:0]	**RES0**	Reserved	**RES0**

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #6, C15, C4, #3{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0100	0b011

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.
SYS #6, C15, C4, #3{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2CDR(X[t]);
```

A.5 AArch64 Identification registers summary

The 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-137: Identification registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
MIDR_EL1	3	0	C0	C0	0	—	64-bit	Main ID Register
MPIDR_EL1	3	0	C0	C0	5	—	64-bit	Multiprocessor Affinity Register
REVIDR_EL1	3	0	C0	C0	6	—	64-bit	Revision ID Register
ID_PFR0_EL1	3	0	C0	C1	0	—	64-bit	AArch32 Processor Feature Register 0
ID_PFR1_EL1	3	0	C0	C1	1	—	64-bit	AArch32 Processor Feature Register 1
ID_DFR0_EL1	3	0	C0	C1	2	—	64-bit	AArch32 Debug Feature Register 0
ID_AFR0_EL1	3	0	C0	C1	3	—	64-bit	AArch32 Auxiliary Feature Register 0
ID_MMFR0_EL1	3	0	C0	C1	4	—	64-bit	AArch32 Memory Model Feature Register 0
ID_MMFR1_EL1	3	0	C0	C1	5	—	64-bit	AArch32 Memory Model Feature Register 1
ID_MMFR2_EL1	3	0	C0	C1	6	—	64-bit	AArch32 Memory Model Feature Register 2
ID_MMFR3_EL1	3	0	C0	C1	7	—	64-bit	AArch32 Memory Model Feature Register 3
ID_ISAR0_EL1	3	0	C0	C2	0	—	64-bit	AArch32 Instruction Set Attribute Register 0
ID_ISAR1_EL1	3	0	C0	C2	1	—	64-bit	AArch32 Instruction Set Attribute Register 1
ID_ISAR2_EL1	3	0	C0	C2	2	—	64-bit	AArch32 Instruction Set Attribute Register 2
ID_ISAR3_EL1	3	0	C0	C2	3	—	64-bit	AArch32 Instruction Set Attribute Register 3
ID_ISAR4_EL1	3	0	C0	C2	4	—	64-bit	AArch32 Instruction Set Attribute Register 4
ID_ISAR5_EL1	3	0	C0	C2	5	—	64-bit	AArch32 Instruction Set Attribute Register 5
ID_MMFR4_EL1	3	0	C0	C2	6	—	64-bit	AArch32 Memory Model Feature Register 4
ID_ISAR6_EL1	3	0	C0	C2	7	—	64-bit	AArch32 Instruction Set Attribute Register 6
MVFR0_EL1	3	0	C0	C3	0	—	64-bit	AArch32 Media and VFP Feature Register 0
MVFR1_EL1	3	0	C0	C3	1	—	64-bit	AArch32 Media and VFP Feature Register 1
MVFR2_EL1	3	0	C0	C3	2	—	64-bit	AArch32 Media and VFP Feature Register 2
ID_PFR2_EL1	3	0	C0	C3	4	—	64-bit	AArch32 Processor Feature Register 2
ID_DFR1_EL1	3	0	C0	C3	5	—	64-bit	Debug Feature Register 1
ID_MMFR5_EL1	3	0	C0	C3	6	—	64-bit	AArch32 Memory Model Feature Register 5
ID_AA64PFR0_EL1	3	0	C0	C4	0	—	64-bit	AArch64 Processor Feature Register 0
ID_AA64PFR1_EL1	3	0	C0	C4	1	—	64-bit	AArch64 Processor Feature Register 1
ID_AA64ZFR0_EL1	3	0	C0	C4	4	—	64-bit	SVE Feature ID register 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ID_AA64DFR0_EL1	3	0	C0	C5	0	–	64-bit	AArch64 Debug Feature Register 0
ID_AA64DFR1_EL1	3	0	C0	C5	1	–	64-bit	AArch64 Debug Feature Register 1
ID_AA64AFR0_EL1	3	0	C0	C5	4	–	64-bit	AArch64 Auxiliary Feature Register 0
ID_AA64AFR1_EL1	3	0	C0	C5	5	–	64-bit	AArch64 Auxiliary Feature Register 1
ID_AA64ISAR0_EL1	3	0	C0	C6	0	–	64-bit	AArch64 Instruction Set Attribute Register 0
ID_AA64ISAR1_EL1	3	0	C0	C6	1	–	64-bit	AArch64 Instruction Set Attribute Register 1
ID_AA64ISAR2_EL1	3	0	C0	C6	2	–	64-bit	AArch64 Instruction Set Attribute Register 2
ID_AA64MMFR0_EL1	3	0	C0	C7	0	–	64-bit	AArch64 Memory Model Feature Register 0
ID_AA64MMFR1_EL1	3	0	C0	C7	1	–	64-bit	AArch64 Memory Model Feature Register 1
ID_AA64MMFR2_EL1	3	0	C0	C7	2	–	64-bit	AArch64 Memory Model Feature Register 2
MPAMIDR_EL1	3	0	C10	C4	4	–	64-bit	MPAM ID Register (EL1)
IMP_CPUCFR_EL1	3	0	C15	C0	0	–	64-bit	CPU Configuration Register
CCSIDR_EL1	3	1	C0	C0	0	–	64-bit	Current Cache Size ID Register
CLIDR_EL1	3	1	C0	C0	1	–	64-bit	Cache Level ID Register
GMID_EL1	3	1	C0	C0	4	–	64-bit	Multiple tag transfer ID register
CSSELR_EL1	3	2	C0	C0	0	–	64-bit	Cache Size Selection Register
CTR_EL0	3	3	C0	C0	1	–	64-bit	Cache Type Register
DCZID_EL0	3	3	C0	C0	7	–	64-bit	Data Cache Zero ID register
VPIDR_EL2	3	4	C0	C0	0	–	64-bit	Virtualization Processor ID Register
VMPIDR_EL2	3	4	C0	C0	5	–	64-bit	Virtualization Multiprocessor ID Register
IMP_CPUMPMMCR_EL3	3	6	C15	C2	1	–	64-bit	Global MPMM Configuration Register

A.5.1 MIDR_EL1, Main ID Register

Provides identification information for the PE, including an implementer code for the device and a device ID number.

Configurations

AArch64 register MIDR_EL1 bits [31:0] are architecturally mapped to External System register B.5.3 MIDR_EL1, Main ID Register on page 654 bits [31:0].

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0100 0001 0000 1111 1101 1000 0000 0010

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-52: AArch64_midr_el1 bit assignments

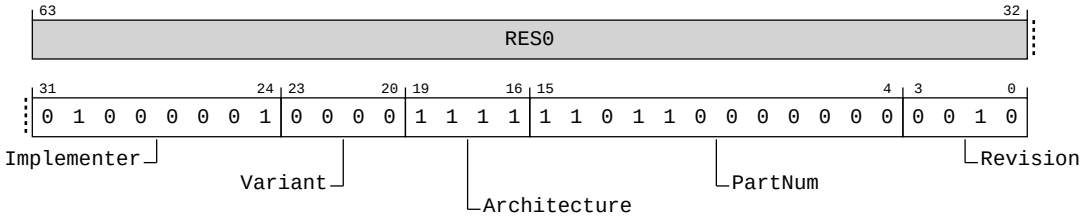


Table A-138: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	Implementer	Indicates the implementer code. This value is: **0b01000001** Arm Limited	0x41
[23:20]	Variant	Indicates the major revision of the product. **0b0000** r0p2	0b0000
[19:16]	Architecture	Architecture version. Defined values are: **0b1111** Architecture is defined by 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. **0b110110000000** Cortex-A520	0xD80
[3:0]	Revision	Indicates the minor revision of the product. **0b0010** r0p2	0b0010

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
        return VPIDR_EL2;
    else
        return MIDR_EL1;
elsif PSTATE.EL == EL2 then
    return MIDR_EL1;
elsif PSTATE.EL == EL3 then
    return MIDR_EL1;
```

A.5.2 MPIDR_EL1, Multiprocessor Affinity Register

In a multiprocessor system, provides an additional PE identification mechanism for scheduling purposes.

Configurations

In a uniprocessor system, Arm recommends that each Aff<n> field of this register returns a value of 0.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

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

Figure A-53: AArch64_mpidr_el1 bit assignments

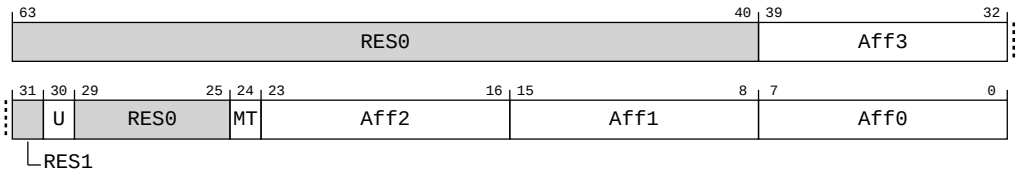


Table A-140: MPIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:40]	**RES0**	Reserved	**RES0**
[39:32]	Aff3	Affinity level 3. See the description of Aff0 for more information. The value will be determined by the CLUSTERIDAFF3 configuration pins.	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.	x
[29:25]	**RES0**	Reserved	**RES0**
[24]	MT	Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. See the description of Aff0 for more information about affinity levels. **0b1** Performance of PEs with different affinity level 0 values, and the same values for affinity level 1 and higher, is very interdependent.	x
[23:16]	Aff2	Affinity level 2. See the description of Aff0 for more information. The value will be determined by the CLUSTERIDAFF2 configuration pins.	8{x}
[15:8]	Aff1	Affinity level 1. See the description of Aff0 for more information. Identification number for each core in an cluster counting from zero.	8{x}
[7:0]	Aff0	Affinity level 0. This is the affinity level that is most significant for determining PE behavior. Higher affinity levels are increasingly less significant in determining PE behavior. The assigned value of the MPIDR.{Aff2, Aff1, Aff0} or AArch64-MPIDR_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole. **0b00000000** Thread 0 Cortex-A520 is single-threaded.	8{x}

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
        return VMPIDR_EL2;
    else
        return MPIDR_EL1;
elsif PSTATE.EL == EL2 then
    return MPIDR_EL1;
elsif PSTATE.EL == EL3 then
    return MPIDR_EL1;
```

A.5.3 REVIDR_EL1, Revision ID Register

Provides implementation-specific minor revision information.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

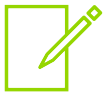
Identification registers

Access type

See bit descriptions

Reset value

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

Figure A-54: AArch64_revidr_el1 bit assignments

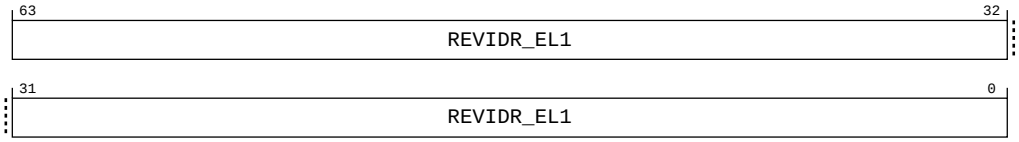


Table A-142: REVIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	REVIDR_EL1	Identifies errata fixes present in this implementation. Refer to the Software Developer's Errata Notice or Product Errata Notice for information on how to interpret this field.	64{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
        return REVIDR_EL1;
elsif PSTATE.EL == EL2 then
    return REVIDR_EL1;
elsif PSTATE.EL == EL3 then
    return REVIDR_EL1;
```

A.5.4 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0

Provides additional information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

The external register ext-EDPFR gives information from this register.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

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

Figure A-55: AArch64_id_aa64pfr0_el1 bit assignments

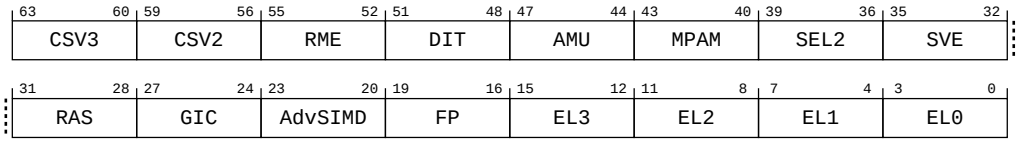


Table A-144: ID_AA64PFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	CSV3	Speculative use of faulting data. Defined values are: **0b0001** Data loaded under speculation with a permission or domain fault cannot be used to form an address, 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.	xxxx
[59:56]	CSV2	Speculative use of out of context branch targets. Defined values are: **0b0010** Branch targets trained in one hardware-described context can exploitatively control speculative execution in a different hardware-described context only in a hard-to-determine way. The SCXTNUM_ELx registers are supported and the contexts include the SCXTNUM_ELx register contexts.	xxxx
[55:52]	RME	Realm Management Extension (RME). Defined values are: **0b0000** Realm Management Extension not implemented.	xxxx

Bits	Name	Description	Reset
[51:48]	DIT	Data Independent Timing. Defined values are: **0b0001** AArch64 provides the PSTATE.DIT mechanism to guarantee constant execution time of certain instructions.	xxxx
[47:44]	AMU	Indicates support for Activity Monitors Extension. Defined values are: **0b0001** FEAT_AMUv1 is implemented.	xxxx
[43:40]	MPAM	Indicates support for MPAM Extension. Defined values are: **0b0001** MPAM Extension version 1.1 is implemented.	xxxx
[39:36]	SEL2	Secure EL2. Defined values are: **0b0001** Secure EL2 is implemented.	xxxx
[35:32]	SVE	Scalable Vector Extension. Defined values are: **0b0001** SVE architectural state and programmers' model are implemented.	xxxx
[31:28]	RAS	RAS Extension version. Defined values are: **0b0010** FEAT_RASv1p1 present.	xxxx
[27:24]	GIC	System register GIC CPU interface. Defined values are: **0b0000** GIC CPU interface system registers not implemented. This value is reported when the GICCDISABLE input is HIGH. **0b0011** System register interface to version 4.1 of the GIC CPU interface is supported. This value is reported when the GICCDISABLE input is LOW.	xxxx
[23:20]	AdvSIMD	Advanced SIMD. Defined values are: **0b0001** Advanced SIMD is implemented, including support for the following SISD and SIMD operations: • Integer byte, halfword, word and doubleword element operations. • Half-precision, single-precision and double-precision floating-point arithmetic. • Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.	xxxx
[19:16]	FP	Floating-point. Defined values are: **0b0001** Floating-point is implemented, and includes support for: • Half-precision, single-precision and double-precision floating-point types. • Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.	xxxx
[15:12]	EL3	EL3 Exception level handling. Defined values are: **0b0001** EL3 can be executed in AArch64 state only.	xxxx

Bits	Name	Description	Reset
[11:8]	EL2	EL2 Exception level handling. Defined values are: **0b0001** EL2 can be executed in AArch64 state only.	xxxx
[7:4]	EL1	EL1 Exception level handling. Defined values are: **0b0001** EL1 can be executed in AArch64 state only.	xxxx
[3:0]	EL0	EL0 Exception level handling. Defined values are: **0b0001** EL0 can be executed in AArch64 state only.	xxxx

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
        return ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64PFR0_EL1;
```

A.5.5 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1

Reserved for future expansion of information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* 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

See bit descriptions

Reset value

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

Figure A-56: AArch64_id_aa64pfr1_el1 bit assignments

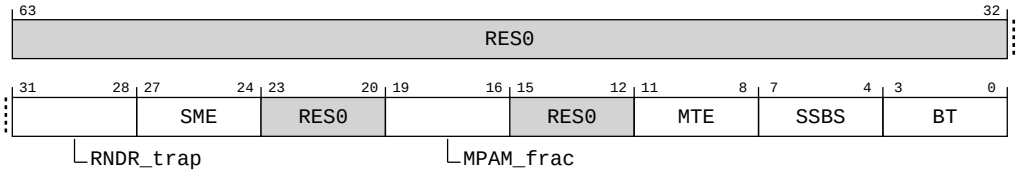


Table A-146: ID_AA64PFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:28]	RNDR_trap	Random Number trap to EL3 field. Defined values are: **0b0000** Trapping of AArch64-RNDR and AArch64-RNDRRS to EL3 is not supported.	xxxx
[27:24]	SME	Scalable Matrix Extension field. Defined values are: **0b0000** SME architectural state and programmers' model are not implemented.	xxxx
[23:20]	**RES0**	Reserved	**RES0**
[19:16]	MPAM_frac	MPAM Extension fractional field. Defined values are: **0b0001** Implements MPAM v1.1 and adds support for MPAM2_EL2.TIDR to provide trapping of MPAMIDR_EL1 when MPAMHCR_EL2 is not present.	xxxx
[15:12]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[11:8]	MTE	Support for the Memory Tagging Extension. Defined values are: **0b0001** Memory Tagging Extension instructions accessible at EL0 are implemented. Instructions and System Registers defined by the extension not configurably accessible at EL0 are Unallocated and other System Register fields defined by the extension are **RES0**. This value is reported when the BROADCASTMTE input is LOW. **0b0011** Memory Tagging Extension is implemented with support for asymmetric Tag Check Fault handling. This value is reported when the BROADCASTMTE input is HIGH.	xxxx
[7:4]	SSBS	Speculative Store Bypassing controls in AArch64 state. Defined values are: **0b0010** AArch64 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypass Safe, and the MSR and MRS instructions to directly read and write the PSTATE.SSBS field.	xxxx
[3:0]	BT	Branch Target Identification mechanism support in AArch64 state. Defined values are: **0b0001** The Branch Target Identification mechanism is implemented.	xxxx

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
        return ID_AA64PFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64PFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64PFR1_EL1;
```

A.5.6 ID_AA64ZFR0_EL1, SVE Feature ID register 0

Provides additional information about the implemented features of the AArch64 Scalable Vector Extension, when the AArch64-ID_AA64PFR0_EL1.SVE field is not zero.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* 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

See bit descriptions

Reset value

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

Figure A-57: AArch64_id_aa64zfr0_el1 bit assignments

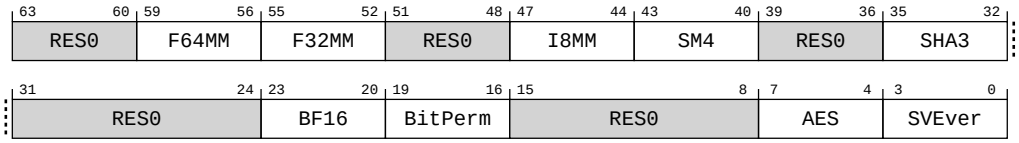


Table A-148: ID_AA64ZFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	RES0	Reserved	RES0
[59:56]	F64MM	Indicates support for SVE FP64 double-precision floating-point matrix multiplication instructions. Defined values are: **0b0000** FP64 matrix multiplication and related instructions are not implemented.	xxxx
[55:52]	F32MM	Indicates support for the SVE FP32 single-precision floating-point matrix multiplication instruction. Defined values are: **0b0000** FP32 matrix multiplication instruction is not implemented.	xxxx

Bits	Name	Description	Reset
[51:48]	**RES0**	Reserved	**RES0**
[47:44]	I8MM	Indicates support for SVE Int8 matrix multiplication instructions. Defined values are: **0b0001** SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	xxxx
[43:40]	SM4	Indicates support for SVE SM4 instructions. Defined values are: **0b0000** SVE2 SM4 instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SVE2 SM4E and SM4EKEY instructions are implemented. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[39:36]	**RES0**	Reserved	**RES0**
[35:32]	SHA3	Indicates support for the SVE SHA3 instructions. Defined values are: **0b0000** SVE2 SHA-3 instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SVE2 RAX1 instruction is implemented. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[31:24]	**RES0**	Reserved	**RES0**
[23:20]	BF16	Indicates support for SVE BFloat16 instructions. Defined values are: **0b0001** BFCVT, BFCVTNT, BFDOT, BFMLALB, BFMLALT, and BFMMLA instructions are implemented.	xxxx
[19:16]	BitPerm	Indicates support for SVE bit permute instructions. Defined values are: **0b0001** SVE BDEP, BEXT, and BGRP instructions are implemented.	xxxx
[15:8]	**RES0**	Reserved	**RES0**
[7:4]	AES	Indicates support for SVE AES instructions. Defined values are: **0b0000** SVE2-AES instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0010** SVE2 AESE, AESD, AESMC, and AESIMC instructions are implemented plus SVE2 PMULLB and PMULLT instructions with 64-bit source. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[3:0]	SVEver	Indicates support for SVE. Defined values are: **0b0001** The SVE and non-optional SVE2 instructions are implemented.	xxxx

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
        return ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ZFR0_EL1;
```

A.5.7 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0

Provides top level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

The external register ext-EDDFR gives information from this register.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

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

Figure A-58: AArch64_id_aa64dfr0_el1 bit assignments

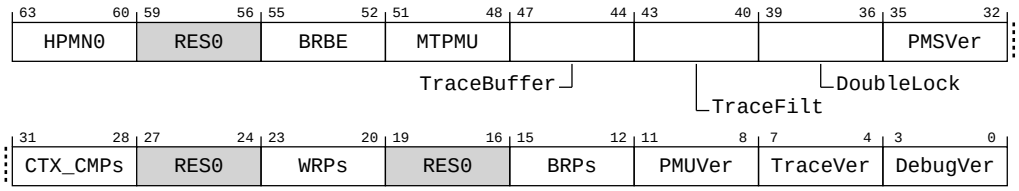


Table A-150: ID_AA64DFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	HPMN0	Zero PMU event counters for a Guest operating system. Defined values are: **0b0001** Setting AArch64-MDCR_EL2.HPMN to zero has defined behavior.	xxxx
[59:56]	**RES0**	Reserved	**RES0**
[55:52]	BRBE	Branch Record Buffer Extension. Defined values are: **0b0000** Branch Record Buffer Extension not implemented.	xxxx
[51:48]	MTPMU	Multi-threaded PMU extension. Defined values are: **0b1111** FEAT_MTPMU not implemented. If FEAT_PMUv3 is implemented, AArch64-PMEVTYPER<n>_EL0.MT and AArch32-PMEVTYPER<n>.MT are RES0.	xxxx
[47:44]	TraceBuffer	Trace Buffer Extension. Defined values are: **0b0001** Trace Buffer Extension implemented, FEAT_TRBE.	xxxx
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. Defined values are: **0b0001** Armv8.4 Self-hosted Trace Extension implemented.	xxxx
[39:36]	DoubleLock	OS Double Lock implemented. Defined values are: **0b1111** OS Double Lock not implemented. AArch64-OSDLR_EL1 is **RAZ/WI**.	xxxx
[35:32]	PMSVer	Statistical Profiling Extension version. Defined values are: **0b0000** Statistical Profiling Extension not implemented.	xxxx
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. **0b0001** Two context-aware breakpoints are included	xxxx
[27:24]	**RES0**	Reserved	**RES0**
[23:20]	WRPs	Number of watchpoints, minus 1. The value of 0b0000 is reserved. **0b0011** Four watchpoints	xxxx
[19:16]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[15:12]	BRPs	Number of breakpoints, minus 1. The value of 0b0000 is reserved. **0b0101** Six breakpoints	xxxx
[11:8]	PMUVer	Performance Monitors Extension version. Defined value is: **0b0111** Performance Monitors Extension implemented, PMUv3 for Armv8.7	xxxx
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a PE trace unit is implemented. Defined values are: **0b0001** PE trace unit System registers implemented.	xxxx
[3:0]	DebugVer	Debug architecture version. Indicates presence of Armv8 debug architecture. Defined values are: **0b1001** Armv8.4 debug architecture, FEAT_Debugv8p4.	xxxx

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
        return ID_AA64DFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64DFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64DFR0_EL1;
```

A.5.8 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1

Reserved for future expansion of top level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* 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

See bit descriptions

Reset value

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

Figure A-59: AArch64_id_aa64dfr1_el1 bit assignments

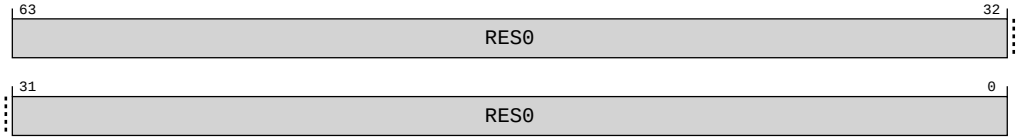


Table A-152: ID_AA64DFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	**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
        return ID_AA64DFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64DFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64DFR1_EL1;
```

A.5.9 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0

Provides information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* 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

See bit descriptions

Reset value

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

Figure A-60: AArch64_id_aa64afr0_el1 bit assignments

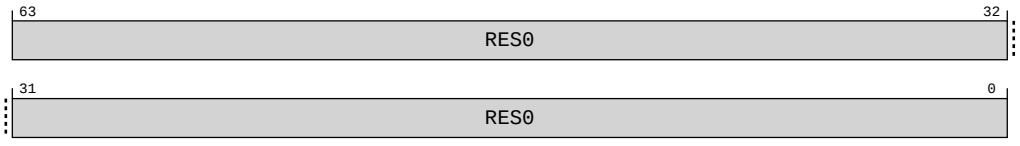


Table A-154: ID_AA64AFR0_EL1 bit descriptions

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
        return ID_AA64AFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64AFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64AFR0_EL1;
```

A.5.10 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1

Reserved for future expansion of information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* 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

See bit descriptions

Reset value

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

Figure A-61: AArch64_id_aa64afr1_el1 bit assignments

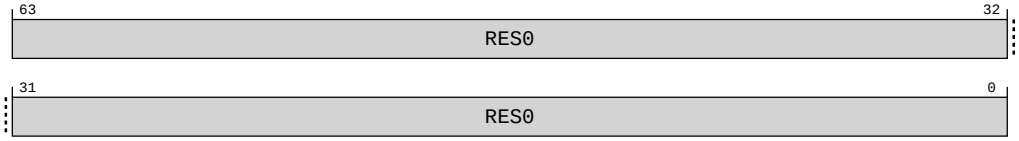


Table A-156: ID_AA64AFR1_EL1 bit descriptions

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
        return ID_AA64AFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64AFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64AFR1_EL1;
```

A.5.11 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0

Provides information about the instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* 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

See bit descriptions

Reset value

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

Figure A-62: AArch64_id_aa64isar0_el1 bit assignments

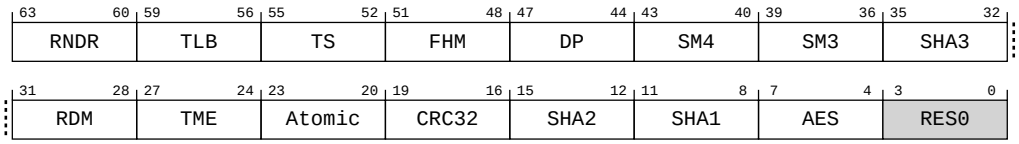


Table A-158: ID_AA64ISAR0_EL1 bit descriptions

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 AArch64-SCR_EL3.TRNDR. Defined values are: **0b0000** No Random Number instructions are implemented.	xxxx
[59:56]	TLB	Indicates support for Outer shareable and TLB range maintenance instructions. Defined values are: **0b0010** Outer shareable and TLB range maintenance instructions are implemented.	xxxx
[55:52]	TS	Indicates support for flag manipulation instructions. Defined values are: **0b0010** CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented.	xxxx

Bits	Name	Description	Reset
[51:48]	FHM	Indicates support for FMLAL and FMLSL instructions. Defined values are: **0b0001** FMLAL and FMLSL instructions are implemented.	xxxx
[47:44]	DP	Indicates support for Dot Product instructions in AArch64 state. Defined values are: **0b0001** UDOT and SDOT instructions implemented.	xxxx
[43:40]	SM4	Indicates support for SM4 instructions in AArch64 state. Defined values are: **0b0000** No SM4 instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SM4E and SM4EKEY instructions implemented. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[39:36]	SM3	Indicates support for SM3 instructions in AArch64 state. Defined values are: **0b0000** No SM3 instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SM3SS1, SM3TT1A, SM3TT1B, SM3TT2A, SM3TT2B, SM3PARTW1, and SM3PARTW2 instructions implemented. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[35:32]	SHA3	Indicates support for SHA3 instructions in AArch64 state. Defined values are: **0b0000** No SHA3 instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** EOR3, RAX1, XAR, and BCAX instructions implemented. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[31:28]	RDM	Indicates support for SQRDMLAH and SQRDMLSH instructions in AArch64 state. Defined values are: **0b0001** SQRDMLAH and SQRDMLSH instructions implemented.	xxxx
[27:24]	TME	Indicates support for TME instructions. Defined values are: **0b0000** TME instructions are not implemented.	xxxx
[23:20]	Atomic	Indicates support for Atomic instructions in AArch64 state. Defined values are: **0b0010** LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions implemented.	xxxx
[19:16]	CRC32	Indicates support for CRC32 instructions in AArch64 state. Defined values are: **0b0001** CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX instructions implemented.	xxxx

Bits	Name	Description	Reset
[15:12]	SHA2	Indicates support for SHA2 instructions in AArch64 state. Defined values are: **0b0000** No SHA2 instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0010** SHA256H, SHA256H2, SHA256SU0, SHA256SU1, SHA512H, SHA512H2, SHA512SU0, and SHA512SU1 instructions implemented. This value is reported when Cryptographic extensions are implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	xxxx
[11:8]	SHA1	Indicates support for SHA1 instructions in AArch64 state. Defined values are: **0b0000** No SHA1 instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions implemented. This value is reported when Cryptographic extensions are implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	xxxx
[7:4]	AES	Indicates support for AES instructions in AArch64 state. Defined values are: **0b0000** No AES instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0010** AESE, AESD, AESMC, and AESIMC instructions are implemented plus PMULL/PMULL2 instructions operating on 64-bit data quantities. This value is reported when Cryptographic extensions are implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	xxxx
[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
        return ID_AA64ISAR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ISAR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ISAR0_EL1;
```

A.5.12 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* 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

See bit descriptions

Reset value

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

Figure A-63: AArch64_id_aa64isar1_el1 bit assignments

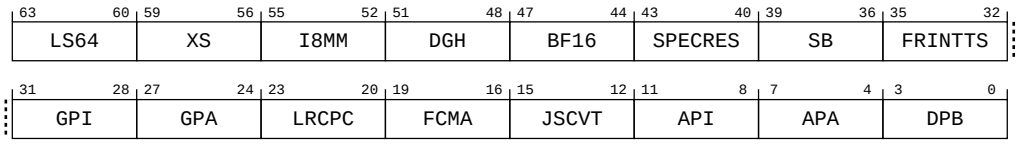


Table A-160: ID_AA64ISAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	LS64	Indicates support for LD64B and ST64B* instructions, and the AArch64-ACCDATA_EL1 register. Defined values of this field are: **0b0000** The LD64B and ST64B* instructions, the AArch64-ACCDATA_EL1 register, and associated traps are not supported.	xxxx
[59:56]	XS	Indicates support for the XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the AArch64-HCRX_EL2.{FGTnXS, FnXS} fields in AArch64 state. Defined values are: **0b0001** The XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the AArch64-HCRX_EL2.{FGTnXS, FnXS} fields are supported.	xxxx
[55:52]	I8MM	Indicates support for Advanced SIMD and Floating-point Int8 matrix multiplication instructions in AArch64 state. Defined values are: **0b0001** SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	xxxx
[51:48]	DGH	Indicates support for the Data Gathering Hint instruction. Defined values are: **0b0001** Data Gathering Hint is implemented.	xxxx
[47:44]	BF16	Indicates support for Advanced SIMD and Floating-point BFloat16 instructions in AArch64 state. Defined values are: **0b0001** BFCVT, BFCVTN, BFCVTN2, BFDOT, BFMLALB, BFMLALT, and BFMMLA instructions are implemented.	xxxx
[43:40]	SPECRES	Indicates support for prediction invalidation instructions in AArch64 state. Defined values are: **0b0001** CFP RCTX, DVP RCTX, and CPP RCTX instructions are implemented.	xxxx
[39:36]	SB	Indicates support for SB instruction in AArch64 state. Defined values are: **0b0001** SB instruction is implemented.	xxxx
[35:32]	FRINTTS	Indicates support for the FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented. Defined values are: **0b0001** FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented.	xxxx
[31:28]	GPI	Indicates support for an **IMPLEMENTATION DEFINED** algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: **0b0000** Generic Authentication using an **IMPLEMENTATION DEFINED** algorithm is not implemented.	xxxx
[27:24]	GPA	Indicates whether the QARMA5 algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: **0b0000** Generic Authentication using the QARMA5 algorithm is not implemented.	xxxx
[23:20]	LRCPC	Indicates support for weaker release consistency, RCpc, based model. Defined values are: **0b0010** The LDAPUR*, STLUR*, and LDAPR* instructions are implemented.	xxxx

Bits	Name	Description	Reset
[19:16]	FCMA	Indicates support for complex number addition and multiplication, where numbers are stored in vectors. Defined values are: **0b0001** The FCMLA and FCADD instructions are implemented.	xxxx
[15:12]	JSCVT	Indicates support for JavaScript conversion from double precision floating point values to integers in AArch64 state. Defined values are: **0b0001** The FJCVTZS instruction is implemented.	xxxx
[11:8]	API	Indicates whether an **IMPLEMENTATION DEFINED** algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the `PACGA` instruction. Defined values are: **0b0000** Address Authentication using an **IMPLEMENTATION DEFINED** algorithm is not implemented.	xxxx
[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. Defined values are: **0b0000** Address Authentication using the QARMA5 algorithm is not implemented.	xxxx
[3:0]	DPB	Data Persistence writeback. Indicates support for the DC CVAP and DC CVADP instructions in AArch64 state. Defined values are: **0b0010** DC CVAP and DC CVADP supported	xxxx

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
        return ID_AA64ISAR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ISAR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ISAR1_EL1;
```

A.5.13 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

See bit descriptions

Reset value

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

Figure A-64: AArch64_id_aa64isar2_el1 bit assignments

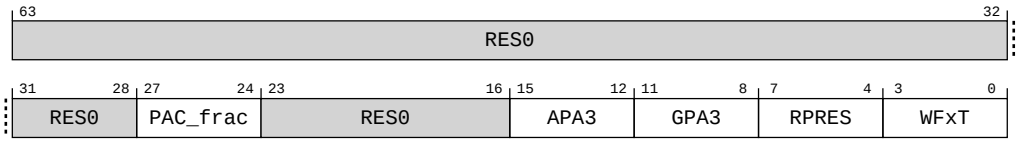


Table A-162: ID_AA64ISAR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:28]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[27:24]	PAC_frac	Indicates whether the ConstPACField() function used as part of the PAC addition returns FALSE or TRUE. **0b0001** ConstPACField() returns TRUE.	xxxx
[23:16]	**RES0**	Reserved	**RES0**
[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. Defined values are: **0b0101** Address Authentication using the QARMA3 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.	xxxx
[11:8]	GPA3	Indicates whether the QARMA3 algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: **0b0001** Generic Authentication using the QARMA3 algorithm is implemented. This includes the `PACGA` instruction.	xxxx
[7:4]	RPRES	When AArch64-FPCR.AH is 1, indicates support for 12 bits of mantissa in reciprocal and reciprocal square root instructions in AArch64 state. Defined values are: **0b0000** Reciprocal and reciprocal square root estimates give 8 bits of mantissa.	xxxx
[3:0]	WFxT	Indicates support for the `WFET` and `WFIT` instructions in AArch64 state. Defined values are: **0b0010** `WFET` and `WFIT` are supported, and the register number is reported in the ESR_ELx on exceptions.	xxxx

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
        return ID_AA64ISAR2_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ISAR2_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ISAR2_EL1;
```

A.5.14 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

See bit descriptions

Reset value

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

Figure A-65: AArch64_id_aa64mmfr0_el1 bit assignments

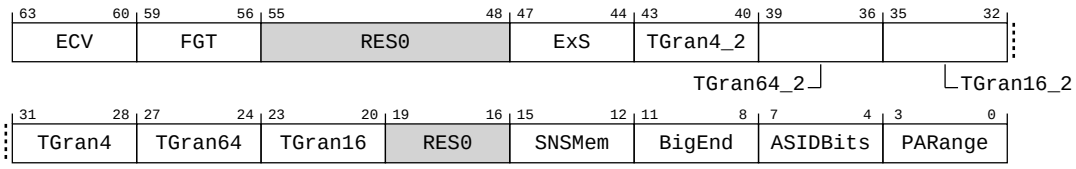


Table A-164: ID_AA64MMFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	ECV	Indicates presence of Enhanced Counter Virtualization. Defined values are: **0b0010** Enhanced Counter Virtualization is implemented. Supports CNTHCTL_EL2.{EL1TVT, EL1TVCT, EL1NVPCT, EL1NVVCT, EVNTIS, ECV}, CNTKCTL_EL1.EVNTIS, CNTPCTSS_EL0 counter views, and CNTVCTSS_EL0 counter views. Extends the PMSCR_EL1.PCT, PMSCR_EL2.PCT, TRFCR_EL1.TS, and TRFCR_EL2.TS fields. Includes support for CNTPOFF_EL2.	xxxx

Bits	Name	Description	Reset
[59:56]	FGT	Indicates presence of the Fine-Grained Trap controls: • If EL2 is implemented, the AArch64-HAFGRTR_EL2, AArch64-HDFGRTR_EL2, AArch64-HDFGWTR_EL2, AArch64-HFGRTR_EL2, AArch64-HFGITR_EL2 and AArch64-HFGWTR_EL2 registers, and their associated traps. • If EL2 is implemented, AArch64-MDCR_EL2.TDCC. • If EL3 is implemented, AArch64-MDCR_EL3.TDCC. • If both EL2 and EL3 are implemented, AArch64-SCR_EL3.FGTEn. Defined values are: **0b0001** The fine-grained trap controls are implemented.	xxxx
[55:48]	**RES0**	Reserved	**RES0**
[47:44]	ExS	Indicates support for disabling context synchronizing exception entry and exit. Defined values are: **0b0000** All exception entries and exits are context synchronization events.	xxxx
[43:40]	TGran4_2	Indicates support for 4KB memory granule size at stage 2. Defined values are: **0b0010** 4KB granule supported at stage 2.	xxxx
[39:36]	TGran64_2	Indicates support for 64KB memory granule size at stage 2. Defined values are: **0b0010** 64KB granule supported at stage 2.	xxxx
[35:32]	TGran16_2	Indicates support for 16KB memory granule size at stage 2. Defined values are: **0b0010** 16KB granule supported at stage 2.	xxxx
[31:28]	TGran4	Indicates support for 4KB memory translation granule size. Defined values are: **0b0000** 4KB granule supported.	xxxx
[27:24]	TGran64	Indicates support for 64KB memory translation granule size. Defined values are: **0b0000** 64KB granule supported.	xxxx
[23:20]	TGran16	Indicates support for 16KB memory translation granule size. Defined values are: **0b0001** 16KB granule supported.	xxxx
[19:16]	**RES0**	Reserved	**RES0**
[15:12]	SNSMem	Indicates support for a distinction between Secure and Non-secure Memory. Defined values are: **0b0001** Does support a distinction between Secure and Non-secure Memory.	xxxx
[11:8]	BigEnd	Indicates support for mixed-endian configuration. Defined values are: **0b0001** Mixed-endian support. The SCTLR_ELx.EE and AArch64-SCTLR_EL1.E0E bits can be configured.	xxxx
[7:4]	ASIDBits	Number of ASID bits. Defined values are: **0b0010** 16 bits.	xxxx

Bits	Name	Description	Reset
[3:0]	PARange	Physical Address range supported. Defined values are: **0b0010** 40 bits, 1TB.	xxxx

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
        return ID_AA64MMFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64MMFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64MMFR0_EL1;
```

A.5.15 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

See bit descriptions

Reset value

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

Figure A-66: AArch64_id_aa64mmfr1_el1 bit assignments

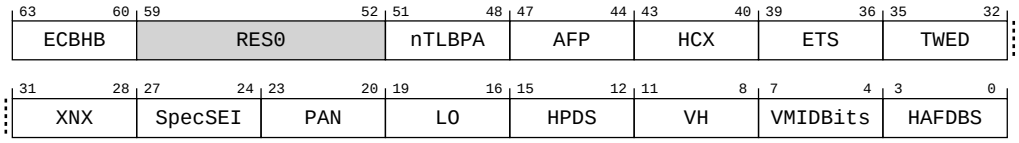


Table A-166: ID_AA64MMFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	ECBHB	Indicates support for mitigation of exploitative control using branch history information between exception levels. Defined values are: **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 code in a different context after the exception.	xxxx
[59:52]	**RES0**	Reserved	**RES0**
[51:48]	nTLBPA	Indicates support for intermediate caching of translation table walks. Defined values are: **0b0001** The intermediate caching of translation table walks does not include non-coherent caches of previous valid translation table entries since the last completed TLBI applicable to the PE where either: • The caching is indexed by the physical address of the location holding the translation table entry. • The caching is used for stage 1 translations and is indexed by the intermediate physical address of the location holding the translation table entry.	xxxx
[47:44]	AFP	Indicates support for AArch64-FPCR.{AH, FIZ, NEP}. Defined values are: **0b0001** The AArch64-FPCR.{AH, FIZ, NEP} fields are supported.	xxxx
[43:40]	HCX	Indicates support for AArch64-HCRX_EL2 and its associated EL3 trap. Defined values are: **0b0001** AArch64-HCRX_EL2 and its associated EL3 trap are supported.	xxxx
[39:36]	ETS	Indicates support for Enhanced Translation Synchronization. Defined values are: **0b0001** Enhanced Translation Synchronization is supported.	xxxx

Bits	Name	Description	Reset
[35:32]	TWED	Indicates support for the configurable delayed trapping of WFE. Defined values are: **0b0000** Configurable delayed trapping of WFE is not supported.	xxxx
[31:28]	XNX	Indicates support for execute-never control distinction by Exception level at stage 2. Defined values are: **0b0001** Distinction between EL0 and EL1 execute-never control at stage 2 supported.	xxxx
[27:24]	SpecSEI	Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches. The defined values of this field are: **0b0001** The PE might generate an SError interrupt due to an External abort on a speculative read.	xxxx
[23:20]	PAN	Privileged Access Never. Indicates support for the PAN bit in PSTATE, AArch64-SPSR_EL1, AArch64-SPSR_EL2, AArch64-SPSR_EL3, and AArch64-DSPSR_EL0. Defined values are: **0b0011** PAN supported, AT S1E1RP and AT S1E1WP instructions supported, and AArch64-SCTLR_EL1.EPAN and AArch64-SCTLR_EL2.EPAN bits supported.	xxxx
[19:16]	LO	LORegions. Indicates support for LORegions. Defined values are: **0b0001** LORegions supported.	xxxx
[15:12]	HPDS	Hierarchical Permission Disables. Indicates support for disabling hierarchical controls in translation tables. Defined values are: **0b0010** Disabling of hierarchical controls supported with the TCR_EL1.{HPD1, HPD0}, TCR_EL2.HPD or TCR_EL2.{HPD1, HPD0}, and TCR_EL3.HPD bits and adds possible hardware allocation of bits[62:59] of the translation table descriptors from the final lookup level for **IMPLEMENTATION DEFINED** use.	xxxx
[11:8]	VH	Virtualization Host Extensions. Defined values are: **0b0001** Virtualization Host Extensions supported.	xxxx
[7:4]	VMIDBits	Number of VMID bits. Defined values are: **0b0010** 16 bits	xxxx
[3:0]	HAFDBS	Hardware updates to Access flag and Dirty state in translation tables. Defined values are: **0b0010** Hardware update of both the Access flag and dirty state is supported.	xxxx

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
        return ID_AA64MMFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64MMFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64MMFR1_EL1;
```

A.5.16 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

See bit descriptions

Reset value

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

Figure A-67: AArch64_id_aa64mmfr2_el1 bit assignments

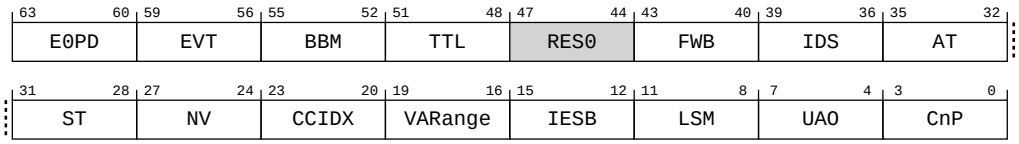


Table A-168: ID_AA64MMFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	E0PD	Indicates support for the E0PD mechanism. Defined values are: **0b0001** E0PDx mechanism is implemented.	xxxx
[59:56]	EVT	Enhanced Virtualization Traps. If EL2 is implemented, indicates support for the AArch64-HCR_EL2.{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps. Defined values are: **0b0010** AArch64-HCR_EL2.{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps are supported.	xxxx
[55:52]	BBM	Allows identification of the requirements of the hardware to have break-before-make sequences when changing block size for a translation. **0b0010** Level 2 support for changing block size is supported.	xxxx
[51:48]	TTL	Indicates support for TTL field in address operations. Defined values are: **0b0001** TLB maintenance instructions by address have bits[47:44] holding the TTL field.	xxxx
[47:44]	**RES0**	Reserved	**RES0**
[43:40]	FWB	Indicates support for AArch64-HCR_EL2.FWB. Defined values are: **0b0001** AArch64-HCR_EL2.FWB is supported.	xxxx
[39:36]	IDS	Indicates the value of ESR_ELx.EC that reports an exception generated by a read access to the feature ID space. Defined values are: **0b0001** All exceptions generated by an AArch64 read access to the feature ID space are reported by ESR_ELx.EC == 0x18.	xxxx
[35:32]	AT	Identifies support for unaligned single-copy atomicity and atomic functions. Defined values are: **0b0001** Unaligned single-copy atomicity and atomic functions with a 16-byte address range aligned to 16-bytes are supported.	xxxx
[31:28]	ST	Identifies support for small translation tables. Defined values are: **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.	xxxx

Bits	Name	Description	Reset
[27:24]	NV	Nested Virtualization. If EL2 is implemented, indicates support for the use of nested virtualization. Defined values are: **0b0000** Nested virtualization is not supported.	xxxx
[23:20]	CCIDX	Support for the use of revised AArch64-CCSIDR_EL1 register format. Defined values are: **0b0001** 64-bit format implemented for all levels of the CCSIDR_EL1.	xxxx
[19:16]	VARange	Indicates support for a larger virtual address. Defined values are: **0b0000** VMSAv8-64 supports 48-bit VAs.	xxxx
[15:12]	IESB	Indicates support for the IESB bit in the SCTLR_ELx registers. Defined values are: **0b0001** IESB bit in the SCTLR_ELx registers is supported.	xxxx
[11:8]	LSM	Indicates support for LSMAOE and nTLSMD bits in AArch64-SCTLR_EL1 and AArch64-SCTLR_EL2. Defined values are: **0b0000** LSMAOE and nTLSMD bits not supported.	xxxx
[7:4]	UAO	User Access Override. Defined values are: **0b0001** UAO supported.	xxxx
[3:0]	CnP	Indicates support for Common not Private translations. Defined values are: **0b0001** Common not Private translations supported.	xxxx

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
        return ID_AA64MMFR2_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64MMFR2_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64MMFR2_EL1;
```

A.5.17 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

See bit descriptions

Reset value

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

MPAMIDR_EL1 indicates the MPAM implementation parameters of the PE.

Figure A-68: AArch64_mpamidr_el1 bit assignments

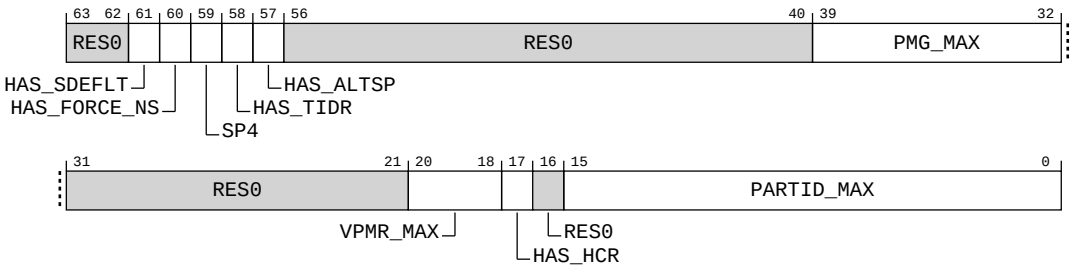


Table A-170: MPAMIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:62]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[61]	HAS_SDEFLT	HAS_SDEFLT indicates support for AArch64-MPAM3_EL3.SDEFLT bit. Defined values are: **0b1** The SDEFLT bit is implemented in AArch64-MPAM3_EL3.	x
[60]	HAS_FORCE_NS	HAS_FORCE_NS indicates support for AArch64-MPAM3_EL3.FORCE_NS bit. Defined values are: **0b0** The FORCE_NS bit is not implemented in AArch64-MPAM3_EL3.	x
[59]	SP4	Supports 4 MPAM PARTID spaces. **0b0** MPAM supports 2 PARTID spaces.	x
[58]	HAS_TIDR	HAS_TIDR indicates support for AArch64-MPAM2_EL2.TIDR bit. Defined values are: **0b1** The TIDR bit is implemented in AArch64-MPAM2_EL2.	x
[57]	HAS_ALTSP	HAS_ALTSP indicates support for alternative PARTID spaces. **0b0** Alternative PARTID spaces are not implemented.	x
[56:40]	**RES0**	Reserved	**RES0**
[39:32]	PMG_MAX	The largest value of PMG that the implementation can generate. The PMG_I and PMG_D fields of every MPAMn_ELx must implement at least enough bits to represent PMG_MAX. **0b00000001** Max PMG field is 1	8{x}
[31:21]	**RES0**	Reserved	**RES0**
[20:18]	VPMR_MAX	Indicates the maximum register index n for the MPAMVPM<n>_EL2 registers. **0b001** 2 MPAMVPMn_EL2 registers are implemented	xxx
[17]	HAS_HCR	HAS_HCR indicates that the PE implementation supports MPAM virtualization, including AArch64-MPAMHCR_EL2, AArch64-MPAMVPMV_EL2, and MPAMVPM<n>_EL2 with n in the range 0 to VPMR_MAX. Must be 0 if EL2 is not implemented in either Security state. **0b1** MPAM virtualization is supported.	x
[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. **0b0000000000111111** Max PARTID field is 63	16{x}

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 Halted() && EDSCR.SDD == '1' 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
        return MPAMIDR_EL1;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMIDR_EL1;
elsif PSTATE.EL == EL3 then
    return MPAMIDR_EL1;
```

A.5.18 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

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-69: AArch64_imp_cpucfr_el1 bit assignments

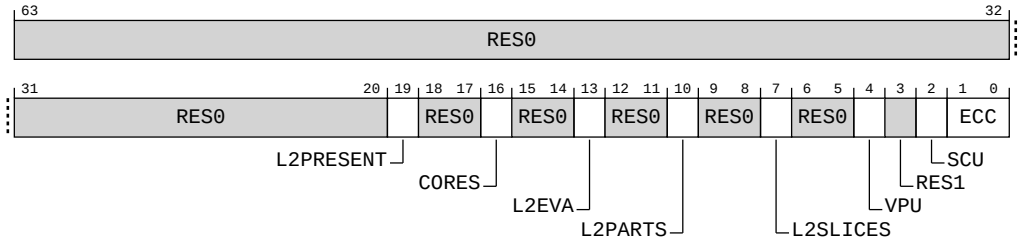


Table A-172: IMP_CPUCFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	**RES0**	Reserved	**RES0**
[19]	L2PRESENT	Indicates whether an L2 cache is present in the complex containing this core. **0b0** An L2 cache is not present in the complex. **0b1** An L2 cache is present in the complex.	x
[18:17]	**RES0**	Reserved	**RES0**
[16]	CORES	The number of cores in the complex containing this core. **0b0** One core. **0b1** Two cores.	x
[15:14]	**RES0**	Reserved	**RES0**
[13]	L2EVA	Indicates whether the L2 cache optimized evict/allocate accesses are implemented. Possible values of this field are: **0b0** Not implemented. **0b1** Implemented.	x
[12:11]	**RES0**	Reserved	**RES0**
[10]	L2PARTS	Indicates the configured number of L2 cache partitions. Possible values of this field are: **0b0** One partition. **0b1** Two partitions.	x
[9:8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7]	L2SLICES	Indicates the configured number of L2 cache slices. Possible values of this field are: **0b0** One slice. **0b1** Two slices.	x
[6:5]	**RES0**	Reserved	**RES0**
[4]	VPU	Describes the configured VPU datapath width. Possible values of this field are: **0b0** Two 64-bit datapaths are configured. **0b1** Two 128-bit datapaths are configured.	x
[3]	**RES1**	Reserved	**RES1**
[2]	SCU	Indicates whether the SCU is present or not. Possible values of this bit are: **0b0** The SCU is present.	x
[1:0]	ECC	Indicates whether ECC is present or not. Possible values of this field are: **0b00** ECC is not present. **0b01** ECC is present.	xx

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
        return IMP_CPUCFR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUCFR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUCFR_EL1;
```

A.5.19 CCSIDR_EL1, Current Cache Size ID Register

Provides information about the architecture of the currently selected cache.

Configurations

The implementation includes one CCSIDR_EL1 for each cache that it can access. AArch64-CSSELR_EL1 selects which Cache Size ID Register is accessible.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

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

Note

The parameters NumSets, Associativity, and LineSize in these registers define the architecturally visible parameters that are required for the cache maintenance by Set/Way instructions. They are not guaranteed to represent the actual microarchitectural features of a design. You cannot make any inference about the actual sizes of caches based on these parameters.

Figure A-70: AArch64_ccsidr_el1 bit assignments

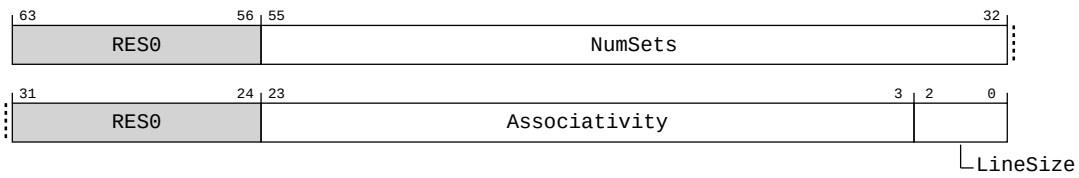


Table A-174: CCSIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[55:32]	NumSets	(Number of sets in cache) - 1, therefore a value of 0 indicates 1 set in the cache. The number of sets does not have to be a power of 2.	24{x}
[31:24]	**RES0**	Reserved	**RES0**
[23:3]	Associativity	(Associativity of cache) - 1, therefore a value of 0 indicates an associativity of 1. The associativity does not have to be a power of 2.	21{x}
[2:0]	LineSize	(Log_2(Number of bytes in cache line)) - 4. For example: • For a line length of 16 bytes: $Log_2(16) = 4$, LineSize entry = 0. This is the minimum line length. • For a line length of 32 bytes: $Log_2(32) = 5$, LineSize entry = 1. When FEAT_MTE2 is implemented and enabled, where a cache only holds Allocation tags, this field is **RES0**.	xxx

Access

If AArch64-CSSELR_EL1.Level is programmed to a cache level that is not implemented, then on a read of the CCSIDR_EL1 the behavior is **CONSTRAINED UNPREDICTABLE**, and can be one of the following:

- The CCSIDR_EL1 read is treated as NOP.
- The CCSIDR_EL1 read is UNDEFINED.
- The CCSIDR_EL1 read returns an **UNKNOWN** value.

MRS <Xt>, CCSIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b001	0b0000	0b0000	0b000

Accessibility

If AArch64-CSSELR_EL1.Level is programmed to a cache level that is not implemented, then on a read of the CCSIDR_EL1 the behavior is CONSTRAINED UNPREDICTABLE, and can be one of the following:

- The CCSIDR_EL1 read is treated as NOP.
- The CCSIDR_EL1 read is UNDEFINED.
- The CCSIDR_EL1 read returns an UNKNOWN value.

MRS <Xt>, CCSIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.CCSIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
```

```
        return CCSIDR_EL1;
elsif PSTATE.EL == EL2 then
    return CCSIDR_EL1;
elsif PSTATE.EL == EL3 then
    return CCSIDR_EL1;
```

A.5.20 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

See bit descriptions

Reset value

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

Figure A-71: AArch64_clidr_el1 bit assignments

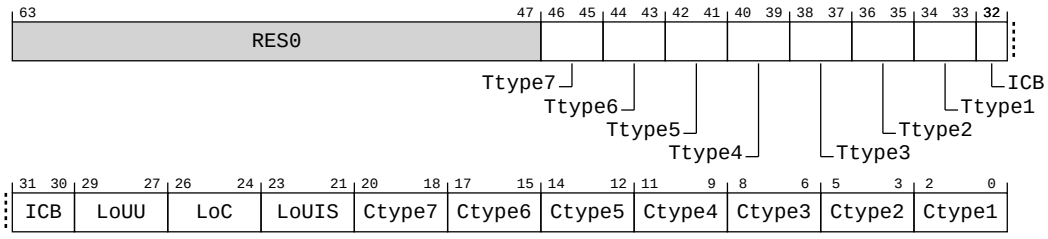


Table A-176: CLIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:47]	**RES0**	Reserved	**RES0**
[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.	xx
[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.	xx
[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.	xx
[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.	xx
[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 is reported if the BROADCASTMTE pin is low or either the Cortex-A520 complex is configured without an L2 cache or the DSU is configured without an L3 cache. **0b10** Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines. This value is reported if the BROADCASTMTE pin is high and both the Cortex-A520 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	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 is reported if the BROADCASTMTE pin is low or both the Cortex-A520 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **0b10** Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines. This value is reported if the BROADCASTMTE pin is high and either the Cortex-A520 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	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 is reported if the BROADCASTMTE pin is low. **0b10** Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines. This value is reported if the BROADCASTMTE pin is high.	xx
[32:30]	ICB	Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions. **0b001** L1 cache is the highest Inner Cacheable level. This value is reported if both the Cortex-A520 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **0b010** L2 cache is the highest Inner Cacheable level. This value is reported if either but not both of the Cortex-A520 complex is configured with an L2 cache or the DSU is configured with an L3 cache. **0b011** L3 cache is the highest Inner Cacheable level. This value is reported if both the Cortex-A520 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	xxx
[29:27]	LoUU	Level of Unification Uniprocessor for the cache hierarchy. **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 D-cache.	xxx
[26:24]	LoC	Level of Coherence for the cache hierarchy. **0b001** Level of Coherency is after the L1 D-cache. This value is reported if both the Cortex-A520 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **0b010** Level of Coherency is after the L2 cache. This value is reported if either but not both of the Cortex-A520 complex is configured with an L2 cache or the DSU is configured with an L3 cache. **0b011** Level of Coherency is after the L3 cache. This value is reported if both the Cortex-A520 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	xxx
[23:21]	LoUIS	Level of Unification Inner Shareable for the cache hierarchy. **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 Inner Shareable is before the L1 D-cache.	xxx

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.	xxx
[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.	xxx
[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.	xxx
[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.	xxx
[8:6]	Ctype3	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache. This value is reported if either the Cortex-A520 complex is configured without an L2 cache or the DSU is configured without an L3 cache. **0b100** Unified cache. This value is reported if both the Cortex-A520 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	xxx
[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: **0b000** No cache. This value is reported if both the Cortex-A520 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **0b100** Unified cache. This value is reported if either the Cortex-A520 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	xxx
[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.	xxx

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
        return CLIDR_EL1;
elsif PSTATE.EL == EL2 then
    return CLIDR_EL1;
elsif PSTATE.EL == EL3 then
    return CLIDR_EL1;
```

A.5.21 GMID_EL1, Multiple tag transfer ID register

Indicates the block size that is accessed by the LDGM and STGM System instructions.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-72: AArch64_gmid_el1 bit assignments

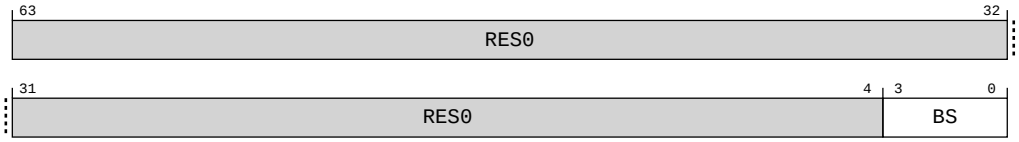


Table A-178: GMID_EL1 bit descriptions

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.	xxxx

Access

MRS <Xt>, GMID_EL1

CRn	op0	op1	op2	CRm
0b0000	0b11	0b001	0b100	0b0000

Accessibility

MRS <Xt>, GMID_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID5 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return GMID_EL1;
elsif PSTATE.EL == EL2 then
    return GMID_EL1;
elsif PSTATE.EL == EL3 then
    return GMID_EL1;
```

A.5.22 CSSELR_EL1, Cache Size Selection Register

Selects the current Cache Size ID Register, AArch64-CCSIDR_EL1, by specifying the required cache level and the cache type (either instruction or data cache).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

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

Figure A-73: AArch64_csselr_el1 bit assignments

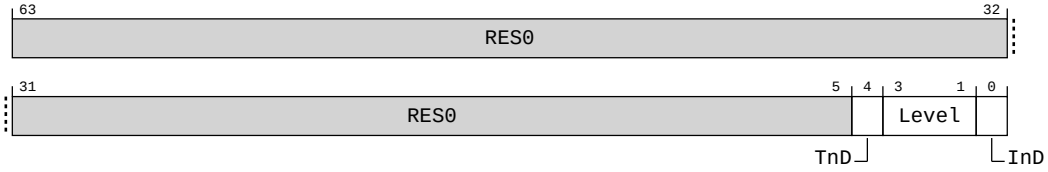


Table A-180: CSSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:5]	RES0	Reserved	RES0
[4]	TnD	Allocation Tag not Data bit. **0b0** Data, Instruction or Unified cache.	x
[3:1]	Level	Cache level of required cache. **0b000** Level 1 cache. **0b001** Level 2 cache. **0b010** Level 3 cache.	xxx

Bits	Name	Description	Reset
[0]	InD	Instruction not Data bit. **0b0** Data or unified cache. **0b1** Instruction cache. If CSSELR_EL1.Level is programmed to a cache level that is not implemented, then a read of CSSELR_EL1 is CONSTRAINED UNPREDICTABLE, and returns **UNKNOWN** values for CSSELR_EL1.{Level, InD}.	x

Access

MRS <Xt>, CSSELR_EL1

op0	op1	CRn	CRm	op2
0b11	0b010	0b0000	0b0000	0b000

MSR CSSELR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b010	0b0000	0b0000	0b000

Accessibility

MRS <Xt>, CSSELR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.CSSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CSSELR_EL1;
elsif PSTATE.EL == EL2 then
    return CSSELR_EL1;
elsif PSTATE.EL == EL3 then
    return CSSELR_EL1;
```

MSR CSSELR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.CSSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        CSSELR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    CSSELR_EL1 = X[t];
```

```
elsif PSTATE.EL == EL3 then
    CSSELR_EL1 = X[t];
```

A.5.23 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

See bit descriptions

Reset value

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

Figure A-74: AArch64_ctr_el0 bit assignments

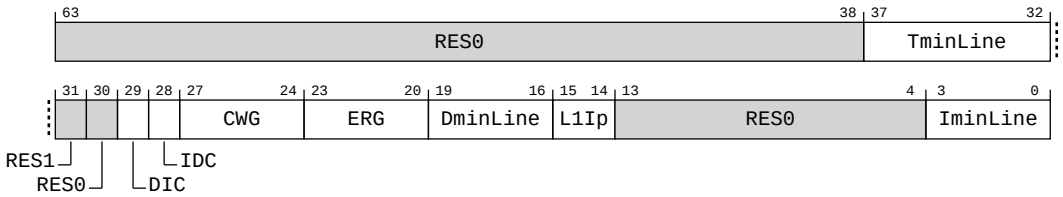


Table A-183: CTR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:38]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[37:32]	TminLine	Tag minimum Line. Log2 of the number of words covered by Allocation Tags in the smallest cache line of all caches which can contain Allocation tags that are controlled by the PE. **0b000000** MTE not supported. This value is reported if the BROADCASTMTE pin is low. **0b000100** 64 bytes. This value is reported if the BROADCASTMTE pin is high.	6{x}
[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.	x
[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.	x
[27:24]	CWG	Cache writeback granule. Log2 of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified. **0b0100** 64 bytes.	xxxx
[23:20]	ERG	Exclusives reservation granule. Log2 of the number of words of the maximum size of the reservation granule for the Load-Exclusive and Store-Exclusive instructions. **0b0100** 64 bytes.	xxxx
[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.	xxxx
[15:14]	L1Ip	Level 1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache. Possible values of this field are: **0b11** Physical Index, Physical Tag (PIPT).	xx
[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.	xxxx

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 !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.UCT == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HFGRTR_EL2.CTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CTR_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.CTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CTR_EL0;
elsif PSTATE.EL == EL2 then
    return CTR_EL0;
elsif PSTATE.EL == EL3 then
    return CTR_EL0;
```

A.5.24 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

See bit descriptions

Reset value

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

Figure A-75: AArch64_dczid_el0 bit assignments

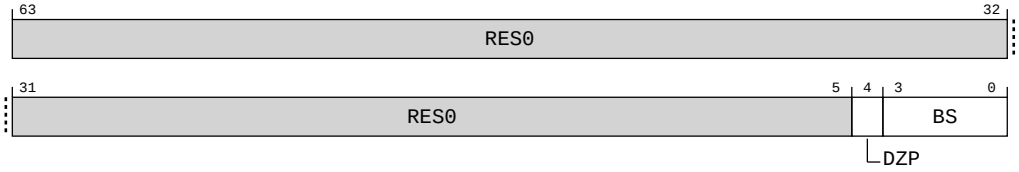


Table A-185: DCZID_EL0 bit descriptions

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. **0b1** Instructions are prohibited. The value read from this field is governed by the access state and the values of the AArch64-HCR_EL2.TDZ and AArch64-SCTLR_EL1.DZE bits.	x
[3:0]	BS	Log_2 of the block size in words. The maximum size supported is 2KB (value == 9). **0b0100** 64 bytes.	xxxx

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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HFGRTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
```

```
        return DCZID_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return DCZID_EL0;
elsif PSTATE.EL == EL2 then
    return DCZID_EL0;
elsif PSTATE.EL == EL3 then
    return DCZID_EL0;
```

A.5.25 IMP_CPUMPMMCR_EL3, Global MPMM Configuration Register

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

Configurations

AArch64 register IMP_CPUMPMMCR_EL3 bits [63:0] are architecturally mapped to External System register B.1.2 CPUMPMMCR, Global MPMM Configuration Register on page 462 bits [63:0].

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-76: AArch64_imp_cpumpmmcr_el3 bit assignments

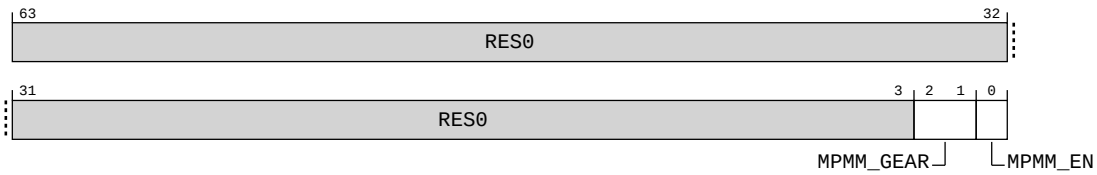


Table A-187: IMP_CPUMPMMCR_EL3 bit descriptions

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.	0b00
[0]	MPMM_EN	MPMM Master 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
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUMPMMCR_EL3;
```

MSR S3_6_C15_C2_1, <Xt>

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

```
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUMPMMCR_EL3 = X[t];
```

A.6 AArch64 GIC system registers summary

The 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-190: GIC system registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICC_PMR_EL1	3	0	C4	C6	0	–	64-bit	Interrupt Controller Interrupt Priority Mask Register
ICV_PMR_EL1	3	0	C4	C6	0	–	64-bit	Interrupt Controller Virtual Interrupt Priority Mask Register
ICC_IAR0_EL1	3	0	C12	C8	0	–	64-bit	Interrupt Controller Interrupt Acknowledge Register 0
ICV_IAR0_EL1	3	0	C12	C8	0	–	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 0
ICC_EOIR0_EL1	3	0	C12	C8	1	–	64-bit	Interrupt Controller End Of Interrupt Register 0
ICV_EOIR0_EL1	3	0	C12	C8	1	–	64-bit	Interrupt Controller Virtual End Of Interrupt Register 0
ICC_HPPIR0_EL1	3	0	C12	C8	2	–	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 0
ICV_HPPIR0_EL1	3	0	C12	C8	2	–	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
ICC_BPR0_EL1	3	0	C12	C8	3	–	64-bit	Interrupt Controller Binary Point Register 0
ICV_BPR0_EL1	3	0	C12	C8	3	–	64-bit	Interrupt Controller Virtual Binary Point Register 0
ICC_AP0R0_EL1	3	0	C12	C8	4	–	64-bit	Interrupt Controller Active Priorities Group 0 Registers
ICV_AP0R0_EL1	3	0	C12	C8	4	–	64-bit	Interrupt Controller Virtual Active Priorities Group 0 Registers
ICC_AP1R0_EL1	3	0	C12	C9	0	–	64-bit	Interrupt Controller Active Priorities Group 1 Registers
ICV_AP1R0_EL1	3	0	C12	C9	0	–	64-bit	Interrupt Controller Virtual Active Priorities Group 1 Registers
ICC_DIR_EL1	3	0	C12	C11	1	–	64-bit	Interrupt Controller Deactivate Interrupt Register
ICV_DIR_EL1	3	0	C12	C11	1	–	64-bit	Interrupt Controller Deactivate Virtual Interrupt Register
ICC_RPR_EL1	3	0	C12	C11	3	–	64-bit	Interrupt Controller Running Priority Register
ICV_RPR_EL1	3	0	C12	C11	3	–	64-bit	Interrupt Controller Virtual Running Priority Register
ICC_SGI1R_EL1	3	0	C12	C11	5	–	64-bit	Interrupt Controller Software Generated Interrupt Group 1 Register
ICC_ASGI1R_EL1	3	0	C12	C11	6	–	64-bit	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
ICC_SGI0R_EL1	3	0	C12	C11	7	–	64-bit	Interrupt Controller Software Generated Interrupt Group 0 Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICC_IAR1_EL1	3	0	C12	C12	0	–	64-bit	Interrupt Controller Interrupt Acknowledge Register 1
ICV_IAR1_EL1	3	0	C12	C12	0	–	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 1
ICC_EOIR1_EL1	3	0	C12	C12	1	–	64-bit	Interrupt Controller End Of Interrupt Register 1
ICV_EOIR1_EL1	3	0	C12	C12	1	–	64-bit	Interrupt Controller Virtual End Of Interrupt Register 1
ICC_HPPIR1_EL1	3	0	C12	C12	2	–	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 1
ICV_HPPIR1_EL1	3	0	C12	C12	2	–	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
ICC_BPR1_EL1	3	0	C12	C12	3	–	64-bit	Interrupt Controller Binary Point Register 1
ICV_BPR1_EL1	3	0	C12	C12	3	–	64-bit	Interrupt Controller Virtual Binary Point Register 1
ICC_CTLR_EL1	3	0	C12	C12	4	–	64-bit	Interrupt Controller Control Register (EL1)
ICV_CTLR_EL1	3	0	C12	C12	4	–	64-bit	Interrupt Controller Virtual Control Register
ICC_SRE_EL1	3	0	C12	C12	5	–	64-bit	Interrupt Controller System Register Enable register (EL1)
ICC_IGRPEN0_EL1	3	0	C12	C12	6	–	64-bit	Interrupt Controller Interrupt Group 0 Enable register
ICV_IGRPEN0_EL1	3	0	C12	C12	6	–	64-bit	Interrupt Controller Virtual Interrupt Group 0 Enable register
ICC_IGRPEN1_EL1	3	0	C12	C12	7	–	64-bit	Interrupt Controller Interrupt Group 1 Enable register
ICV_IGRPEN1_EL1	3	0	C12	C12	7	–	64-bit	Interrupt Controller Virtual Interrupt Group 1 Enable register
ICH_AP0R0_EL2	3	4	C12	C8	0	–	64-bit	Interrupt Controller Hyp Active Priorities Group 0 Registers
ICH_AP1R0_EL2	3	4	C12	C9	0	–	64-bit	Interrupt Controller Hyp Active Priorities Group 1 Registers
ICC_SRE_EL2	3	4	C12	C9	5	–	64-bit	Interrupt Controller System Register Enable register (EL2)
ICH_HCR_EL2	3	4	C12	C11	0	–	64-bit	Interrupt Controller Hyp Control Register
ICH_VTR_EL2	3	4	C12	C11	1	–	64-bit	Interrupt Controller VGIC Type Register
ICH_MISR_EL2	3	4	C12	C11	2	–	64-bit	Interrupt Controller Maintenance Interrupt State Register
ICH_EISR_EL2	3	4	C12	C11	3	–	64-bit	Interrupt Controller End of Interrupt Status Register
ICH_ELRSR_EL2	3	4	C12	C11	5	–	64-bit	Interrupt Controller Empty List Register Status Register
ICH_VMCR_EL2	3	4	C12	C11	7	–	64-bit	Interrupt Controller Virtual Machine Control Register
ICH_LR0_EL2	3	4	C12	C12	0	–	64-bit	Interrupt Controller List Registers
ICH_LR1_EL2	3	4	C12	C12	1	–	64-bit	Interrupt Controller List Registers
ICH_LR2_EL2	3	4	C12	C12	2	–	64-bit	Interrupt Controller List Registers
ICH_LR3_EL2	3	4	C12	C12	3	–	64-bit	Interrupt Controller List Registers
ICC_CTLR_EL3	3	6	C12	C12	4	–	64-bit	Interrupt Controller Control Register (EL3)
ICC_SRE_EL3	3	6	C12	C12	5	–	64-bit	Interrupt Controller System Register Enable register (EL3)
ICC_IGRPEN1_EL3	3	6	C12	C12	7	–	64-bit	Interrupt Controller Interrupt Group 1 Enable register (EL3)

A.6.1 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Registers

Provides information about Group 0 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Figure A-77: AArch64_icc_ap0r0_el1 bit assignments

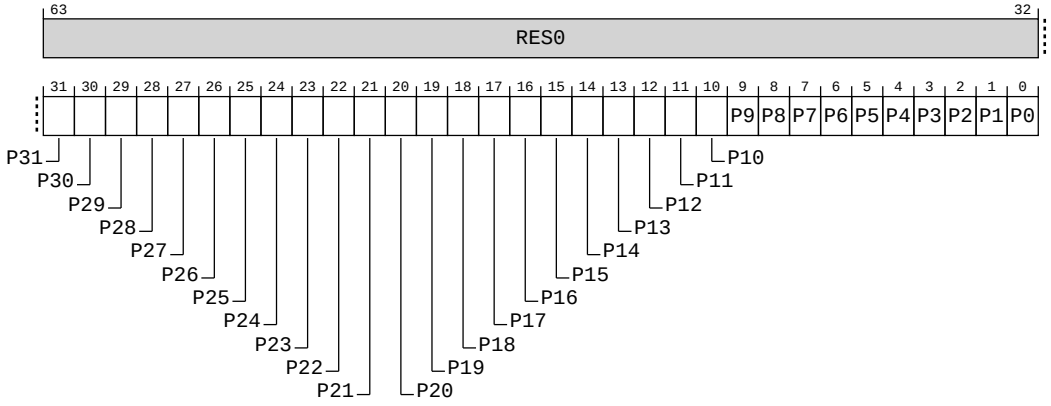


Table A-191: ICC_AP0R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	P<x>	Provides the access to the active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	32{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

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 interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC_AP0R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICC_AP0R2_EL1 and ICC_AP0R3_EL1 are only implemented in implementations that support 7 or more bits of priority. Unimplemented registers are **UNDEFINED**.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits.

Writing to the active priority registers in any order other than the following order will result in **UNPREDICTABLE** behavior:

- ICC_AP0R<n>_EL1.
- Secure AArch64-ICC_AP1R<n>_EL1.
- Non-secure AArch64-ICC_AP1R<n>_EL1.

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 interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC_AP0R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICC_AP0R2_EL1 and ICC_AP0R3_EL1 are only implemented in implementations that support 7 or more bits of priority. Unimplemented registers are UNDEFINED.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- ICC_AP0R<n>_EL1.
- Secure AArch64-ICC_AP1R<n>_EL1.
- Non-secure AArch64-ICC_AP1R<n>_EL1.

MRS <Xt>, ICC_AP0R0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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
        return ICV_AP0R0_EL1;
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_AP0R0_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_AP0R0_EL1;
elsif PSTATE.EL == EL3 then
    return ICC_AP0R0_EL1;
```

MSR ICC_AP0R0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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_AP0R0_EL1 = X[t];
    elsif SCR_EL3.FIQ == '1' then
```

```
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ICC_AP0R0_EL1 = X[t];
```

A.6.2 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers

Provides information about virtual Group 0 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-78: AArch64_icv_ap0r0_el1 bit assignments

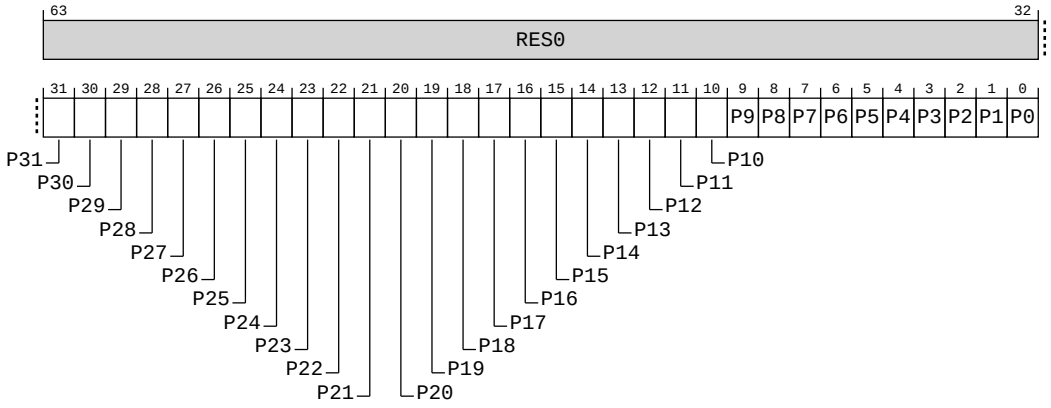


Table A-194: ICV_AP0R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	P<x>	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	32{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

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 only implemented in implementations that support 6 or more bits of priority. ICV_AP0R2_EL1 and ICV_AP0R3_EL1 are only implemented 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_AP0R<n>_EL1.
- AArch64-ICV_AP1R<n>_EL1.

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 only implemented in implementations that support 6 or more bits of priority. ICV_AP0R2_EL1 and ICV_AP0R3_EL1 are only implemented 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_AP0R<n>_EL1.
- AArch64-ICV_AP1R<n>_EL1.

MRS <Xt>, ICC_AP0R0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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
        return ICV_AP0R0_EL1;
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_AP0R0_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_AP0R0_EL1;
elsif PSTATE.EL == EL3 then
    return ICC_AP0R0_EL1;
```

MSR ICC_AP0R0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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_AP0R0_EL1 = X[t];
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ICC_AP0R0_EL1 = X[t];
```

A.6.3 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Registers

Provides information about Group 1 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Figure A-79: AArch64_icc_ap1r0_el1 bit assignments

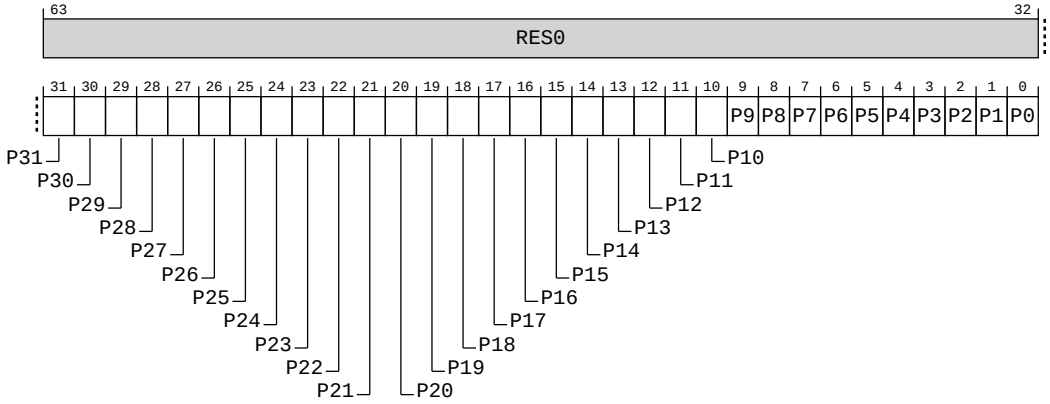


Table A-197: ICC_AP1R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	P<x>	Group 1 interrupt active priorities. When AArch64-SCR_EL3.NS == '1', accesses the priorities for Non-secure Group 1 interrupts, and when AArch64-SCR_EL3.NS == '0' accesses the priorities for Secure Group 1 interrupts. Possible values of each bit are: **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]. When accessed from non-secure EL2 or EL1, only the 16 lowest-priority interrupts are visible in bits [15:0] of this register.	32{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

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 interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC_AP1R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICC_AP1R2_EL1 and ICC_AP1R3_EL1 are only implemented in implementations that support 7 or more bits of priority. Unimplemented registers are **UNDEFINED**.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits.

Writing to the active priority registers in any order other than the following order will result in **UNPREDICTABLE** behavior:

- AArch64-ICC_AP0R<n>_EL1.
- Secure ICC_AP1R<n>_EL1.
- Non-secure ICC_AP1R<n>_EL1.

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 interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC_AP1R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICC_AP1R2_EL1 and ICC_AP1R3_EL1 are only implemented in implementations that support 7 or more bits of priority. Unimplemented registers are UNDEFINED.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- AArch64-ICC_AP0R<n>_EL1.
- Secure ICC_AP1R<n>_EL1.
- Non-secure ICC_AP1R<n>_EL1.

MRS <Xt>, ICC_AP1R0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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
        return ICV_AP1R0_EL1;
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_AP1R0_EL1_S;
        else
            return ICC_AP1R0_EL1_NS;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_AP1R0_EL1_S;
        else
            return ICC_AP1R0_EL1_NS;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        return ICC_AP1R0_EL1_S;
    else
        return ICC_AP1R0_EL1_NS;
```

MSR ICC_AP1R0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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_AP1R0_EL1 = X[t];
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R0_EL1_S = X[t];
        else
            ICC_AP1R0_EL1_NS = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R0_EL1_S = X[t];
        else
            ICC_AP1R0_EL1_NS = X[t];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        ICC_AP1R0_EL1_S = X[t];
    else
        ICC_AP1R0_EL1_NS = X[t];
```

A.6.4 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers

Provides information about virtual Group 1 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Figure A-80: AArch64_icv_ap1r0_el1 bit assignments

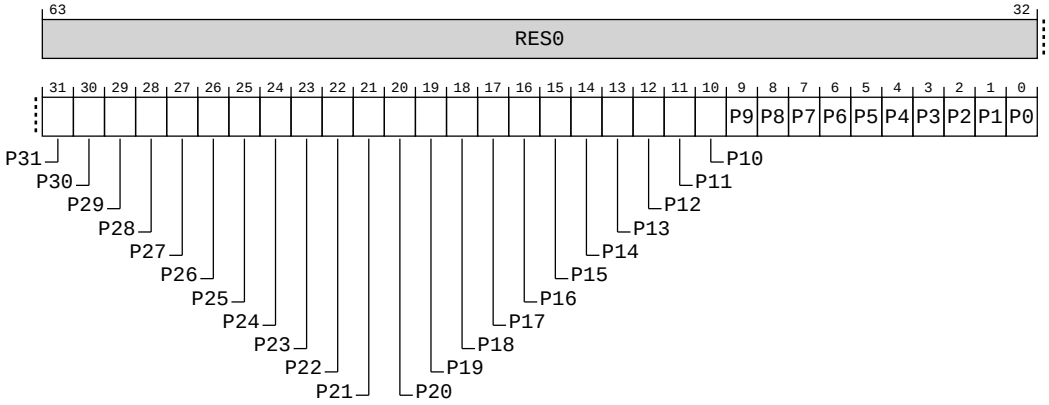


Table A-200: ICV_AP1R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	P<x>	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].	32{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

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 only implemented in implementations that support 6 or more bits of priority. ICV_AP1R2_EL1 and ICV_AP1R3_EL1 are only implemented 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:

- AArch64-ICV_AP0R<n>_EL1.
- ICV_AP1R<n>_EL1.

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 only implemented in implementations that support 6 or more bits of priority. ICV_AP1R2_EL1 and ICV_AP1R3_EL1 are only implemented 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:

- AArch64-ICV_AP0R<n>_EL1.
- ICV_AP1R<n>_EL1.

MRS <Xt>, ICC_AP1R0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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
        return ICV_AP1R0_EL1;
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_AP1R0_EL1_S;
        else
            return ICC_AP1R0_EL1_NS;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_AP1R0_EL1_S;
        else
            return ICC_AP1R0_EL1_NS;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        return ICC_AP1R0_EL1_S;
    else
        return ICC_AP1R0_EL1_NS;
```

MSR ICC_AP1R0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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_AP1R0_EL1 = X[t];
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R0_EL1_S = X[t];
        else
            ICC_AP1R0_EL1_NS = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R0_EL1_S = X[t];
        else
            ICC_AP1R0_EL1_NS = X[t];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        ICC_AP1R0_EL1_S = X[t];
    else
        ICC_AP1R0_EL1_NS = X[t];
```

A.6.5 ICC_CTLR_EL1, Interrupt Controller Control Register (EL1)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Figure A-81: AArch64_icc_ctlr_el1 bit assignments

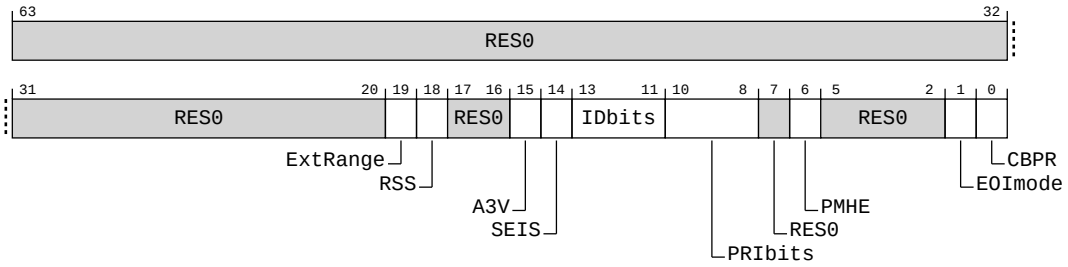


Table A-203: ICC_CTLR_EL1 bit descriptions

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

Bits	Name	Description	Reset
[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 non-zero 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
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits). An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits). **Note:** This field always returns the number of priority bits implemented, regardless of the Security state of the access or the value of ext-GICD_CTLR.DS. For physical accesses, this field determines the minimum value of AArch64-ICC_BPR0_EL1. If EL3 is implemented, physical accesses return the value from AArch64-ICC_CTLR_EL3.PRIbits. **0b100** 5 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 AArch64-ICC_PMR_EL1 as a hint for interrupt distribution. **0b1** Enables use of AArch64-ICC_PMR_EL1 as a hint for interrupt distribution. If EL3 is implemented, this bit is an alias of AArch64-ICC_CTLR_EL3.PMHE. Whether this bit can be written as part of an access to this register depends on the value of ext-GICD_CTLR.DS: • If ext-GICD_CTLR.DS == 0, this bit is read-only. • If ext-GICD_CTLR.DS == 1, this bit is read/write.	x
[5:2]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[1]	EOImode	EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt: **0b0** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality. The Secure AArch64-ICC_CTLR_EL1.EOImode is an alias of AArch64-ICC_CTLR_EL3.EOImode_EL1S. The Non-secure AArch64-ICC_CTLR_EL1.EOImode is an alias of AArch64-ICC_CTLR_EL3.EOImode_EL1NS	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** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. AArch64-ICC_BPR1_EL1 determines the preemption group for Group 1 interrupts. **0b1** AArch64-ICC_BPR0_EL1 determines the preemption group for both Group 0 and Group 1 interrupts. If EL3 is implemented: • This bit is an alias of AArch64-ICC_CTLR_EL3.CBPR_EL1{S,NS} where S or NS corresponds to the current Security state. • If ext-GICD_CTLR.DS == 0, this bit is read-only. • If ext-GICD_CTLR.DS == 1, this bit is read/write.	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 Halted() && EDSCR.SDD == '1' && 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
```

```
        return ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        return ICV_CTLR_EL1;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        return ICC_CTLR_EL1_S;
    else
        return ICC_CTLR_EL1_NS;
```

MSR ICC_CTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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];
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
```

```
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        ICC_CTLR_EL1_S = X[t];
    else
        ICC_CTLR_EL1_NS = X[t];
```

A.6.6 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register

Controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Figure A-82: AArch64_icv_ctlr_el1 bit assignments

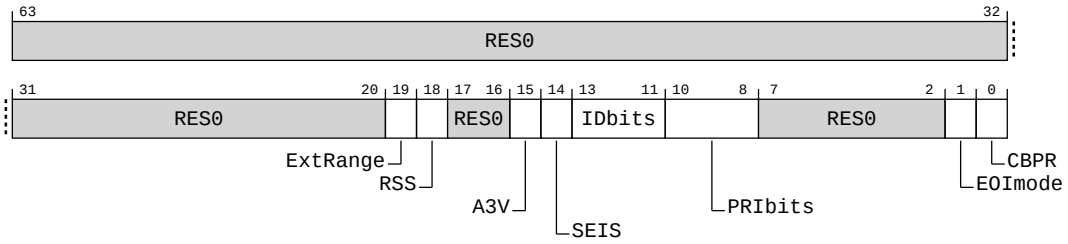


Table A-206: ICV_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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 virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.	x
[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.	x
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of virtual 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 must implement at least 32 levels of physical priority (5 priority bits). **Note:** This field always returns the number of priority bits implemented. The division between group priority and subpriority is defined in the binary point registers AArch64-ICV_BPR0_EL1 and AArch64-ICV_BPR1_EL1. **0b100** 5 bits of priority are implemented	xxx
[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** AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICV_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide priority drop functionality only. AArch64-ICV_DIR_EL1 provides interrupt deactivation functionality.	x

Bits	Name	Description	Reset
[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** AArch64-ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts. **0b1** Non-secure reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1 plus one, saturated to 0b111. Non-secure writes to AArch64-ICV_BPR1_EL1 are ignored. Secure reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1. Secure writes of AArch64-ICV_BPR1_EL1 modify AArch64-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 Halted() && EDSCR.SDD == '1' && 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
        return ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        return ICV_CTLR_EL1;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
```

```
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        return ICC_CTLR_EL1_S;
    else
        return ICC_CTLR_EL1_NS;
```

MSR ICC_CTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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];
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        ICC_CTLR_EL1_S = X[t];
    else
        ICC_CTLR_EL1_NS = X[t];
```

A.6.7 ICH_VTR_EL2, Interrupt Controller VGIC Type Register

Reports supported GIC virtualization features.

Configurations

If EL2 is not implemented, all bits in this register are RES0 from EL3, except for nV4, which is RES1 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Figure A-83: AArch64_ich_vtr_el2 bit assignments

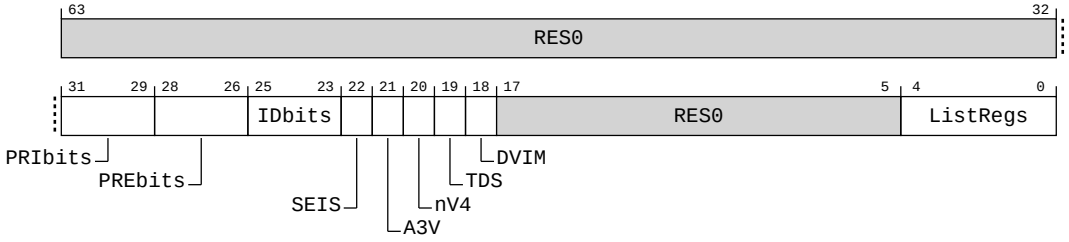


Table A-209: ICH_VTR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:29]	PRIbits	Priority bits. The number of virtual priority bits implemented, minus one. An implementation must implement at least 32 levels of virtual priority (5 priority bits). This field is an alias of AArch64-ICV_CTLR_EL1.PRIbits. **0b100** 5 virtual priority bits are implemented	xxx

Bits	Name	Description	Reset
[28:26]	PREbits	The number of virtual preemption bits implemented, minus one. An implementation must implement at least 32 levels of virtual preemption priority (5 preemption bits). The value of this field must be less than or equal to the value of ICH_VTR_EL2.PRIbits. The maximum value of this field is 6, indicating 7 bits of preemption. This field determines the minimum value of AArch64-ICH_VMCR_EL2.VBPR0. **0b100** 5 virtual pre-emption bits are implemented	xxx
[25:23]	IDbits	The number of virtual interrupt identifier bits supported: **0b000** 16 bits.	xxx
[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.	x
[21]	A3V	Affinity 3 Valid. Possible values are: **0b1** The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.	x
[20]	nV4	Direct injection of virtual interrupts not supported. Possible values are: **0b0** The CPU interface logic supports direct injection of virtual interrupts.	x
[19]	TDS	Separate trapping of EL1 writes to AArch64-ICV_DIR_EL1 supported. **0b1** Implementation supports AArch64-ICH_HCR_EL2.TDIR.	x
[18]	DVIM	Masking of directly-injected virtual interrupts. **0b0** Masking of Directly-injected Virtual Interrupts not supported.	x
[17:5]	**RES0**	Reserved	**RES0**
[4:0]	ListRegs	The number of implemented List registers, minus one. For example, a value of 0b01111 indicates that the maximum of 16 List registers are implemented. **0b00011** Four list registers are implemented.	5{x}

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
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return ICH_VTR_EL2;
elsif PSTATE.EL == EL3 then
    return ICH_VTR_EL2;
```

A.6.8 ICC_CTLR_EL3, Interrupt Controller Control Register (EL3)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x0xx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-84: AArch64_icc_ctlr_el3 bit assignments

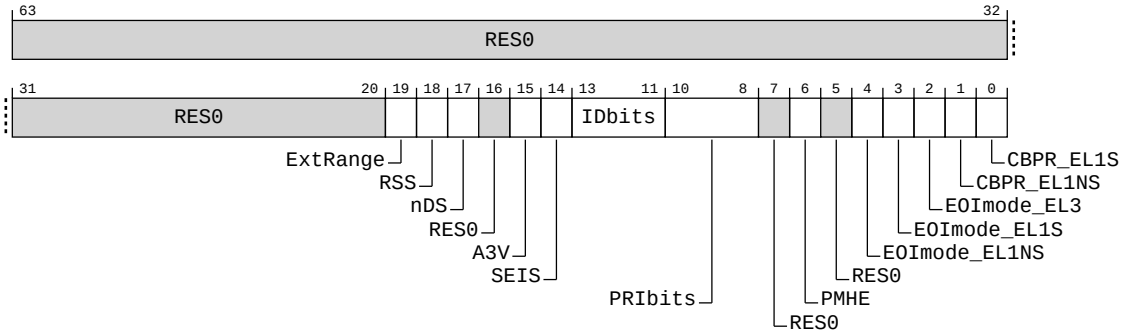


Table A-211: ICC_CTLR_EL3 bit descriptions

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
[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 non-zero 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

Bits	Name	Description	Reset
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits). An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits). **Note:** This field always returns the number of priority bits implemented, regardless of the value of SCR_EL3.NS or the value of ext-GICD_CTLR.DS. The division between group priority and subpriority is defined in the binary point registers AArch64-ICC_BPR0_EL1 and AArch64-ICC_BPR1_EL1. This field determines the minimum value of ICC_BPR0_EL1. **0b100** 5 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. Software must write AArch64-ICC_PMR_EL1 to 0xFF before clearing this field to 0. • An implementation might choose to make this field **RAO/WI** if priority-based routing is always used • An implementation might choose to make this field RAZ/WI if priority-based routing is never used If EL3 is present, AArch64-ICC_CTLR_EL1.PMHE is an alias of ICC_CTLR_EL3.PMHE.	0b0
[5]	**RES0**	Reserved	**RES0**
[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** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality. If EL3 is present, AArch64-ICC_CTLR_EL1(NS).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1NS.	x

Bits	Name	Description	Reset
[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** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality. If EL3 is present, AArch64-ICC_CTLR_EL1(S).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1S.	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** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-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** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. AArch64-ICC_BPR1_EL1 determines the preemption group for Non-secure Group 1 interrupts. **0b1** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Non-secure Group 1 interrupts. Non-secure accesses to ext-GICC_BPR and AArch64-ICC_BPR1_EL1 access the state of AArch64-ICC_BPR0_EL1. If EL3 is present, AArch64-ICC_CTLR_EL1(NS).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1NS.	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** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. AArch64-ICC_BPR1_EL1 determines the preemption group for Secure Group 1 interrupts. **0b1** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Secure Group 1 interrupts. Secure EL1 accesses to AArch64-ICC_BPR1_EL1 access the state of AArch64-ICC_BPR0_EL1. If EL3 is present, AArch64-ICC_CTLR_EL1(S).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1S.	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
    return ICC_CTLR_EL3;
```

MSR ICC_CTLR_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    ICC_CTLR_EL3 = X[t];
```

A.7 AArch64 Performance Monitors registers summary

The summary table provides an overview of all Performance Monitors 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-214: Performance Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMINTENSET_EL1	3	0	C9	C14	1	—	64-bit	Performance Monitors Interrupt Enable Set register
PMINTENCLR_EL1	3	0	C9	C14	2	—	64-bit	Performance Monitors Interrupt Enable Clear register
PMMIR_EL1	3	0	C9	C14	6	—	64-bit	Performance Monitors Machine Identification Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMCR_EL0	3	3	C9	C12	0	–	64-bit	Performance Monitors Control Register
PMCNTENSET_EL0	3	3	C9	C12	1	–	64-bit	Performance Monitors Count Enable Set register
PMCNTENCLR_EL0	3	3	C9	C12	2	–	64-bit	Performance Monitors Count Enable Clear register
PMOVSCLR_EL0	3	3	C9	C12	3	–	64-bit	Performance Monitors Overflow Flag Status Clear Register
PMSWINC_EL0	3	3	C9	C12	4	–	64-bit	Performance Monitors Software Increment register
PMSELR_EL0	3	3	C9	C12	5	–	64-bit	Performance Monitors Event Counter Selection Register
PMCEID0_EL0	3	3	C9	C12	6	–	64-bit	Performance Monitors Common Event Identification register 0
PMCEID1_EL0	3	3	C9	C12	7	–	64-bit	Performance Monitors Common Event Identification register 1
PMCCNTR_EL0	3	3	C9	C13	0	–	64-bit	Performance Monitors Cycle Count Register
PMXEVTYPER_EL0	3	3	C9	C13	1	–	64-bit	Performance Monitors Selected Event Type Register
PMXEVCNTR_EL0	3	3	C9	C13	2	–	64-bit	Performance Monitors Selected Event Count Register
PMUSERENR_EL0	3	3	C9	C14	0	–	64-bit	Performance Monitors User Enable Register
PMOVSSET_EL0	3	3	C9	C14	3	–	64-bit	Performance Monitors Overflow Flag Status Set register
PMEVCNTR0_EL0	3	3	C14	C8	0	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR1_EL0	3	3	C14	C8	1	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR2_EL0	3	3	C14	C8	2	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR3_EL0	3	3	C14	C8	3	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR4_EL0	3	3	C14	C8	4	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR5_EL0	3	3	C14	C8	5	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR6_EL0	3	3	C14	C8	6	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR7_EL0	3	3	C14	C8	7	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR8_EL0	3	3	C14	C9	0	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR9_EL0	3	3	C14	C9	1	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR10_EL0	3	3	C14	C9	2	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR11_EL0	3	3	C14	C9	3	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR12_EL0	3	3	C14	C9	4	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR13_EL0	3	3	C14	C9	5	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR14_EL0	3	3	C14	C9	6	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR15_EL0	3	3	C14	C9	7	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR16_EL0	3	3	C14	C10	0	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR17_EL0	3	3	C14	C10	1	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR18_EL0	3	3	C14	C10	2	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR19_EL0	3	3	C14	C10	3	–	64-bit	Performance Monitors Event Count Registers
PMEVTYPER0_EL0	3	3	C14	C12	0	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER1_EL0	3	3	C14	C12	1	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER2_EL0	3	3	C14	C12	2	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER3_EL0	3	3	C14	C12	3	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER4_EL0	3	3	C14	C12	4	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER5_EL0	3	3	C14	C12	5	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER6_EL0	3	3	C14	C12	6	–	64-bit	Performance Monitors Event Type Registers

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMEVTYPER7_EL0	3	3	C14	C12	7	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER8_EL0	3	3	C14	C13	0	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER9_EL0	3	3	C14	C13	1	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER10_EL0	3	3	C14	C13	2	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER11_EL0	3	3	C14	C13	3	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER12_EL0	3	3	C14	C13	4	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER13_EL0	3	3	C14	C13	5	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER14_EL0	3	3	C14	C13	6	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER15_EL0	3	3	C14	C13	7	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER16_EL0	3	3	C14	C14	0	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER17_EL0	3	3	C14	C14	1	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER18_EL0	3	3	C14	C14	2	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER19_EL0	3	3	C14	C14	3	–	64-bit	Performance Monitors Event Type Registers
PMCCFILTR_EL0	3	3	C14	C15	7	–	64-bit	Performance Monitors Cycle Count Filter Register

A.7.1 PMMIR_EL1, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation to software.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0110 0000 0010 0000 0011

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-85: AArch64_pmmir_el1 bit assignments

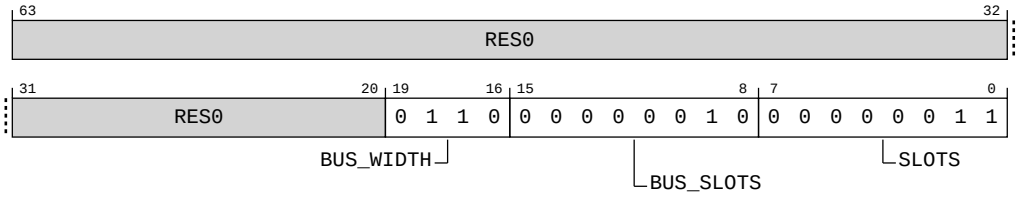


Table A-215: PMMIR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	RES0
[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. Defined values are: **0b0110** 32 bytes.	0b0110
[15:8]	BUS_SLOTS	Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle. **0b00000010** The largest value by which the BUS_ACCESS PMU event may increment in one cycle is 2.	0x02
[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. **0b00000011** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 3.	0x03

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    else
        return PMMIR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMMIR_EL1;
elsif PSTATE.EL == EL3 then
    return PMMIR_EL1;
```

A.7.2 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 [7:0] are architecturally mapped to External System register B.4.27 PMCR_EL0, Performance Monitors Control Register on page 606 bits [7:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 xxxx xxxx xxxx xxxx xxx0 x000


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-86: AArch64_pmcr_el0 bit assignments

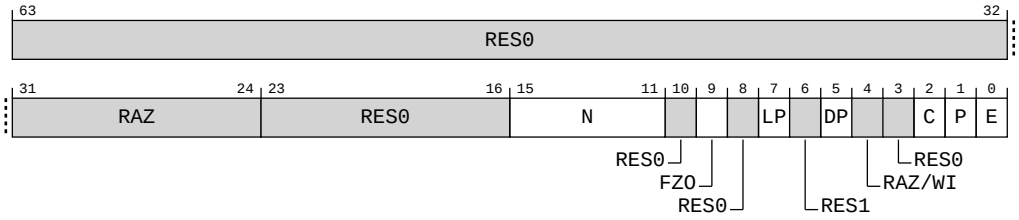


Table A-217: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[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 AArch64-PMCCNTR_EL0 is implemented. If the value is `0b11111`, then AArch64-PMCCNTR_EL0 and 31 event counters are implemented. When EL2 is implemented and enabled for the current Security state, reads of this field from EL1 and EL0 return the value of AArch64-MDCR_EL2.HPMN. **`0b10100`** Twenty PMU Counters Implemented **`0b00110`** Six PMU Counters Implemented	The reset values can be the following: `0b10100`, `0b00110`, respective to the value.
[10]	**RES0**	Reserved	**RES0**
[9]	FZO	Freeze-on-overflow. Stop event counters on overflow. In the description of this field: • If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, PMN is PMCR_EL0.N. **`0b0`** Do not freeze on overflow. **`0b1`** Event counter AArch64-PMEVCNTR<n>_EL0 does not count when AArch64-PMOVSCLR_EL0[(PMN-1):0] is nonzero and n is in the range of affected event counters. If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)]. This field does not affect the operation of other event counters and AArch64-PMCCNTR_EL0. The operation of this field applies even when EL2 is disabled in the current Security state.	`x`
[8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7]	LP	Long event counter enable. Determines when unsigned overflow is recorded by an event counter overflow bit. In the description of this field: • If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, PMN is PMCR_EL0.N. **0b0** Event counter overflow on increment that causes unsigned overflow of AArch64-PMEVCNTR<n>_EL0[31:0]. **0b1** Event counter overflow on increment that causes unsigned overflow of AArch64-PMEVCNTR<n>_EL0[63:0]. If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)]. This field does not affect the operation of other event counters and AArch64-PMCCNTR_EL0. The operation of this field applies even when EL2 is disabled in the current Security state.	x
[6]	**RES1**	Reserved	**RES1**
[5]	DP	Disable cycle counter when event counting is prohibited. **0b0** Cycle counting by AArch64-PMCCNTR_EL0 is not affected by this mechanism. **0b1** Cycle counting by AArch64-PMCCNTR_EL0 is disabled in prohibited regions: • If FEAT_PMUv3p1 is implemented, EL2 is implemented, and AArch64-MDCR_EL2.HPMD is 1, then cycle counting by AArch64-PMCCNTR_EL0 is disabled at EL2. • If FEAT_PMUv3p7 is implemented, EL3 is implemented and using AArch64, and AArch64-MDCR_EL3.MPMX is 1, then cycle counting by AArch64-PMCCNTR_EL0 is disabled at EL3. • If EL3 is implemented, AArch64-MDCR_EL3.SPME is 0, and either FEAT_PMUv3p7 is not implemented or AArch64-MDCR_EL3.MPMX is 0, then cycle counting by AArch64-PMCCNTR_EL0 is disabled at EL3 and in Secure state. If AArch64-MDCR_EL2.HPMN is not 0, this is when event counting by event counters in the range [0..(AArch64-MDCR_EL2.HPMN-1)] is prohibited. For more information see *Prohibiting event counting* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[4]	**RAZ/WI**	Reserved	**RAZ/WI**
[3]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[2]	C	Cycle counter reset. The effects of writing to this bit are: **0b0** No action. **0b1** Reset AArch64-PMCCNTR_EL0 to zero. **Note:** Resetting AArch64-PMCCNTR_EL0 does not change the cycle counter overflow bit. If FEAT_PMUv3p5 is implemented, the value of PMCR_EL0.LC is ignored, and bits [63:0] of the cycle counter are reset. Access to this field is: WO/**RAZ**	0b0
[1]	P	Event counter reset. In the description of this field: • If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, PMN is PMCR_EL0.N. **0b0** No action. **0b1** If n is in the range of affected event counters, resets each event counter AArch64-PMEVCNTR<n> to zero. The effects of writing to this bit are: • If EL2 is implemented and enabled in the current Security state, in EL0 and EL1, if PMN is not 0, a write of 1 to this bit resets event counters in the range [0 .. (PMN-1)]. • If EL2 is disabled in the current Security state, a write of 1 to this bit resets all the event counters. • In EL2 and EL3, a write of 1 to this bit resets all the event counters. • This field does not affect the operation of other event counters and AArch64-PMCCNTR_EL0. **Note:** Resetting the event counters does not change the event counter overflow bits. If FEAT_PMUv3p5 is implemented, the values of AArch64-MDCR_EL2.HLP and PMCR_EL0.LP are ignored, and bits [63:0] of all affected event counters are reset. Access to this field is: WO/**RAZ**	0b0

Bits	Name	Description	Reset
[0]	E	Enable. If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. If EL2 is not implemented, PMN is PMCR_EL0.N. **0b0** AArch64-PMCCNTR_EL0 is disabled and event counters AArch64-PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are disabled. **0b1** AArch64-PMCCNTR_EL0 and event counters AArch64-PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are enabled by AArch64-PMCNTENSET_EL0. If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)]. This field does not affect the operation of other event counters. The operation of this field applies even when EL2 is disabled in the current Security state.	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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCR_EL0;
elsif PSTATE.EL == EL1 then
```

```
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCR_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCR_EL0;
elsif PSTATE.EL == EL3 then
    return PMCR_EL0;
```

MSR PMCR_EL0, <Xt>

```
if PSTATE.EL == EL0 then
    if Halted() && EDSCR.SDD == '1' && 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() && HCR_EL2.<E2H,TGE> != '11' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t];
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && MDCR_EL3.TPM == '1' then
```

```
        UNDEFINED;
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t];
elsif PSTATE.EL == EL3 then
    PMCR_EL0 = X[t];
```

A.7.3 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 System register B.4.28 PMCEID0, Performance Monitors Common Event Identification register 0 on page 610 bits [31:0].

AArch64 register PMCEID0_EL0 bits [63:32] are architecturally mapped to External System register B.4.30 PMCEID2, Performance Monitors Common Event Identification register 2 on page 619 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Figure A-87: AArch64_pmceid0_el0 bit assignments

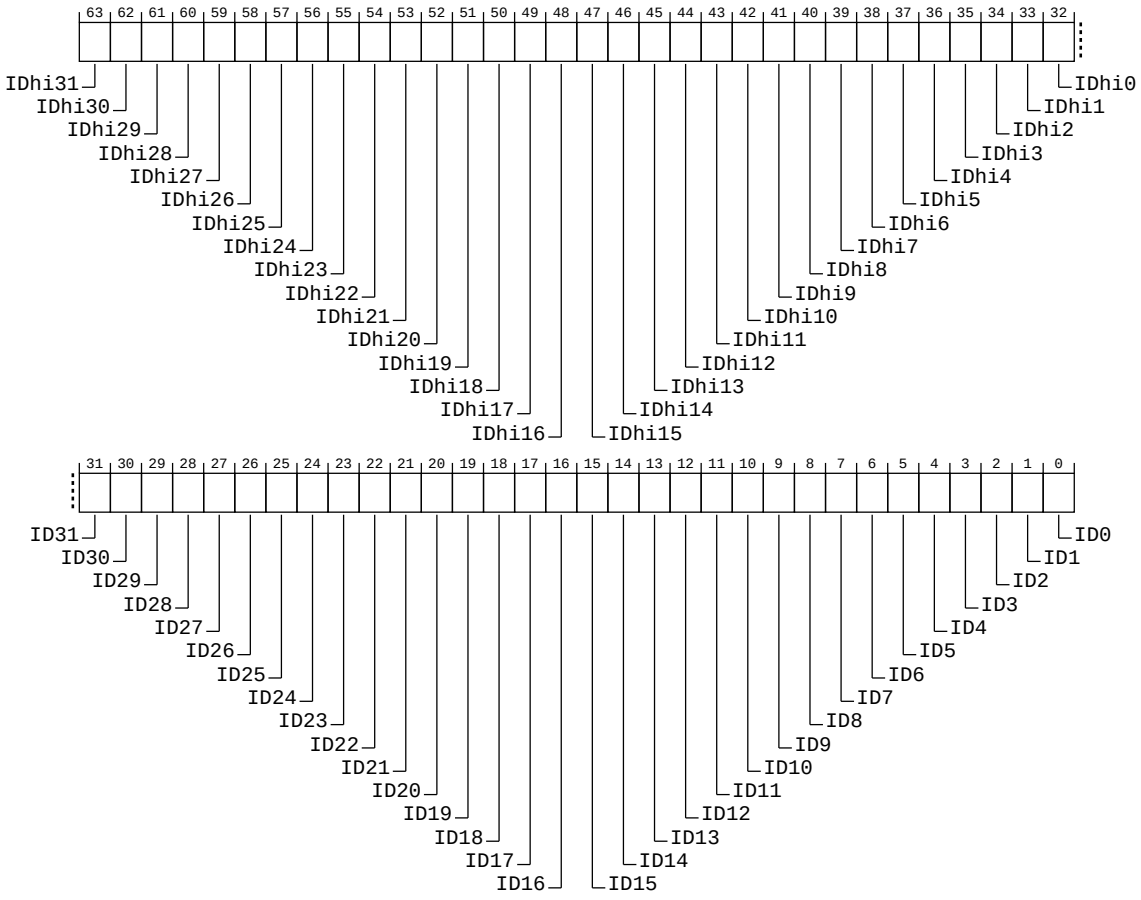


Table A-220: PMCEID0_EL0 bit descriptions

Bits	Name	Description	Reset
[63]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x401f) **0b0** The Common event is not implemented, or not counted.	x
[62]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x401e) **0b0** The Common event is not implemented, or not counted.	x
[61]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x401d) **0b0** The Common event is not implemented, or not counted.	x
[60]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x401c) **0b0** The Common event is not implemented, or not counted.	x

Bits	Name	Description	Reset
[59]	IDhi27	IDhi27 corresponds to common event (0x401b) CTI_TRIGOUT7 **0b1** The Common event is implemented.	x
[58]	IDhi26	IDhi26 corresponds to common event (0x401a) CTI_TRIGOUT6 **0b1** The Common event is implemented.	x
[57]	IDhi25	IDhi25 corresponds to common event (`0x4019`) CTI_TRIGOUT5 **0b1** The Common event is implemented.	x
[56]	IDhi24	IDhi24 corresponds to common event (`0x4018`) CTI_TRIGOUT4 **0b1** The Common event is implemented.	x
[55]	IDhi23	IDhi23 corresponds to a Reserved Event event (`0x4017`) **0b0** The Common event is not implemented, or not counted.	x
[54]	IDhi22	IDhi22 corresponds to a Reserved Event event (`0x4016`) **0b0** The Common event is not implemented, or not counted.	x
[53]	IDhi21	IDhi21 corresponds to a Reserved Event event (`0x4015`) **0b0** The Common event is not implemented, or not counted.	x
[52]	IDhi20	IDhi20 corresponds to a Reserved Event event (`0x4014`) **0b0** The Common event is not implemented, or not counted.	x
[51]	IDhi19	IDhi19 corresponds to common event (`0x4013`) TRCEXTOUT3 **0b1** The Common event is implemented.	x
[50]	IDhi18	IDhi18 corresponds to common event (`0x4012`) TRCEXTOUT2 **0b1** The Common event is implemented.	x
[49]	IDhi17	IDhi17 corresponds to common event (`0x4011`) TRCEXTOUT1 **0b1** The Common event is implemented.	x
[48]	IDhi16	IDhi16 corresponds to common event (`0x4010`) TRCEXTOUT0 **0b1** The Common event is implemented.	x
[47]	IDhi15	IDhi15 corresponds to common event (0x400f) PMU_HOVFS **0b0** The Common event is not implemented, or not counted.	x
[46]	IDhi14	IDhi14 corresponds to common event (0x400e) TRB_TRIG **0b1** The Common event is implemented.	x

Bits	Name	Description	Reset
[45]	IDhi13	IDhi13 corresponds to common event (0x400d) PMU_OVFS **0b0** The Common event is not implemented, or not counted.	x
[44]	IDhi12	IDhi12 corresponds to common event (0x400c) TRB_WRAP **0b1** The Common event is implemented.	x
[43]	IDhi11	IDhi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD **0b0** The common event is not implemented, or not counted. This value is reported if either the Cortex-A520 complex is configured without an L2 cache or the DSU is configured without an L3 cache. **0b1** The common event is implemented. This value is reported if both the Cortex-A520 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	x
[42]	IDhi10	IDhi10 corresponds to common event (0x400a) L2I_CACHE_LMISS **0b0** The Common event is not implemented, or not counted.	x
[41]	IDhi9	IDhi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD **0b0** The common event is not implemented, or not counted. This value is reported if both the Cortex-A520 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **0b1** The common event is implemented. This value is reported if either the Cortex-A520 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	x
[40]	IDhi8	IDhi8 corresponds to common event (0x4008) Reserved **0b0** The Common event is not implemented, or not counted.	x
[39]	IDhi7	IDhi7 corresponds to common event (0x4007) Reserved **0b0** The Common event is not implemented, or not counted.	x
[38]	IDhi6	IDhi6 corresponds to common event (0x4006) L1I_CACHE_LMISS **0b1** The Common event is implemented.	x
[37]	IDhi5	IDhi5 corresponds to common event (0x4005) STALL_BACKEND_MEM **0b1** The Common event is implemented.	x
[36]	IDhi4	IDhi4 corresponds to common event (0x4004) CNT_CYCLES **0b0** The Common event is not implemented, or not counted.	x
[35]	IDhi3	IDhi3 corresponds to common event (0x4003) SAMPLE_COLLISION **0b0** The Common event is not implemented, or not counted.	x

Bits	Name	Description	Reset
[34]	IDhi2	IDhi2 corresponds to common event (0x4002) SAMPLE_FILTRATE **0b0** The Common event is not implemented, or not counted.	x
[33]	IDhi1	IDhi1 corresponds to common event (0x4001) SAMPLE_FEED **0b0** The Common event is not implemented, or not counted.	x
[32]	IDhi0	IDhi0 corresponds to common event (0x4000) SAMPLE_POP **0b0** The Common event is not implemented, or not counted.	x
[31]	ID31	ID31 corresponds to common event (0x1f) L1D_CACHE_ALLOCATE **0b0** The Common event is not implemented, or not counted.	x
[30]	ID30	ID30 corresponds to common event (0x1e) CHAIN **0b1** The Common event is implemented.	x
[29]	ID29	ID29 corresponds to common event (0x1d) BUS_CYCLES **0b1** The Common event is implemented.	x
[28]	ID28	ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED **0b1** The Common event is implemented.	x
[27]	ID27	ID27 corresponds to common event (0x1b) INST_SPEC **0b1** The Common event is implemented.	x
[26]	ID26	ID26 corresponds to common event (0x1a) MEMORY_ERROR **0b1** The Common event is implemented.	x
[25]	ID25	ID25 corresponds to common event (0x19) BUS_ACCESS **0b1** The Common event is implemented.	x
[24]	ID24	ID24 corresponds to common event (0x18) L2D_CACHE_WB **0b0** The common event is not implemented, or not counted. This value is reported if the Cortex-A520 complex is configured without an L2 cache. **0b1** The common event is implemented. This value is reported if the Cortex-A520 complex is configured with an L2 cache.	x

Bits	Name	Description	Reset
[23]	ID23	ID23 corresponds to common event (0x17) L2D_CACHE_REFILL **0b0** The common event is not implemented, or not counted. This value is reported if the Cortex-A520 complex is configured without an L2 cache. **0b1** The common event is implemented. This value is reported if the Cortex-A520 complex is configured with an L2 cache.	x
[22]	ID22	ID22 corresponds to common event (0x16) L2D_CACHE **0b0** The common event is not implemented, or not counted. This value is reported if both the Cortex-A520 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **0b1** The common event is implemented. This value is reported if either the Cortex-A520 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	x
[21]	ID21	ID21 corresponds to common event (0x15) L1D_CACHE_WB **0b1** The Common event is implemented.	x
[20]	ID20	ID20 corresponds to common event (0x14) L1I_CACHE **0b1** The Common event is implemented.	x
[19]	ID19	ID19 corresponds to common event (0x13) MEM_ACCESS **0b1** The Common event is implemented.	x
[18]	ID18	ID18 corresponds to common event (0x12) BR_PRED **0b1** The Common event is implemented.	x
[17]	ID17	ID17 corresponds to common event (0x11) CPU_CYCLES **0b1** The Common event is implemented.	x
[16]	ID16	ID16 corresponds to common event (0x10) BR_MIS_PRED **0b1** The Common event is implemented.	x
[15]	ID15	ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED **0b0** The Common event is not implemented, or not counted.	x
[14]	ID14	ID14 corresponds to common event (0xe) BR_RETURN_RETIRED **0b1** The Common event is implemented.	x
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED **0b1** The Common event is implemented.	x

Bits	Name	Description	Reset
[12]	ID12	ID12 corresponds to common event (0xc) PC_WRITE_RETIRED **0b1** The Common event is implemented.	x
[11]	ID11	ID11 corresponds to common event (0xb) CID_WRITE_RETIRED **0b1** The Common event is implemented.	x
[10]	ID10	ID10 corresponds to common event (0xa) EXC_RETURN **0b1** The Common event is implemented.	x
[9]	ID9	ID9 corresponds to common event (0x9) EXC_TAKEN **0b1** The Common event is implemented.	x
[8]	ID8	ID8 corresponds to common event (0x8) INST_RETIRED **0b1** The Common event is implemented.	x
[7]	ID7	ID7 corresponds to common event (0x7) ST_RETIRED **0b1** The Common event is implemented.	x
[6]	ID6	ID6 corresponds to common event (0x6) LD_RETIRED **0b1** The Common event is implemented.	x
[5]	ID5	ID5 corresponds to common event (0x5) L1D_TLB_REFILL **0b1** The Common event is implemented.	x
[4]	ID4	ID4 corresponds to common event (0x4) L1D_CACHE **0b1** The Common event is implemented.	x
[3]	ID3	ID3 corresponds to common event (0x3) L1D_CACHE_REFILL **0b1** The Common event is implemented.	x
[2]	ID2	ID2 corresponds to common event (0x2) L1I_TLB_REFILL **0b1** The Common event is implemented.	x
[1]	ID1	ID1 corresponds to common event (0x1) L1I_CACHE_REFILL **0b1** The Common event is implemented.	x
[0]	ID0	ID0 corresponds to common event (0x0) SW_INCR **0b1** The Common event is implemented.	x

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 Halted() && EDSCR.SDD == '1' && 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() && HCR_EL2.<E2H,TGE> != '11' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID0_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID0_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID0_EL0;
elsif PSTATE.EL == EL3 then
    return PMCEID0_EL0;
```

A.7.4 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 System register B.4.29 PMCEID1, Performance Monitors Common Event Identification register 1 on page 614 bits [31:0].

AArch64 register PMCEID1_EL0 bits [63:32] are architecturally mapped to External System register B.4.31 PMCEID3, Performance Monitors Common Event Identification register 3 on page 623 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Figure A-88: AArch64_pmceid1_el0 bit assignments

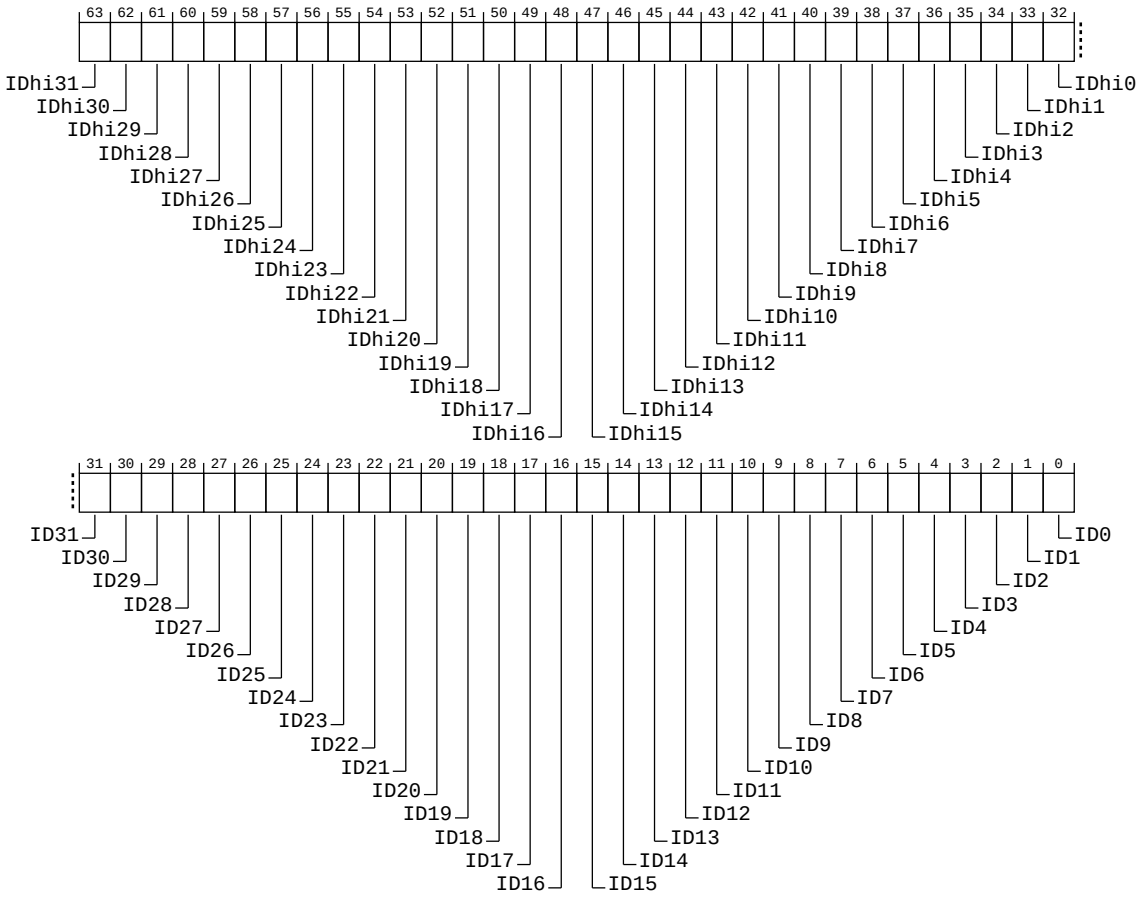


Table A-222: PMCEID1_EL0 bit descriptions

Bits	Name	Description	Reset
[63]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x403f) **0b0** The Common event is not implemented, or not counted.	x
[62]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x403e) **0b0** The Common event is not implemented, or not counted.	x
[61]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x403d) **0b0** The Common event is not implemented, or not counted.	x
[60]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x403c) **0b0** The Common event is not implemented, or not counted.	x

Bits	Name	Description	Reset
[59]	IDhi27	IDhi27 corresponds to a Reserved Event event (0x403b) **0b0** The Common event is not implemented, or not counted.	x
[58]	IDhi26	IDhi26 corresponds to a Reserved Event event (0x403a) **0b0** The Common event is not implemented, or not counted.	x
[57]	IDhi25	IDhi25 corresponds to a Reserved Event event (0x4039) **0b0** The Common event is not implemented, or not counted.	x
[56]	IDhi24	IDhi24 corresponds to a Reserved Event event (0x4038) **0b0** The Common event is not implemented, or not counted.	x
[55]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4037) **0b0** The Common event is not implemented, or not counted.	x
[54]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4036) **0b0** The Common event is not implemented, or not counted.	x
[53]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4035) **0b0** The Common event is not implemented, or not counted.	x
[52]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4034) **0b0** The Common event is not implemented, or not counted.	x
[51]	IDhi19	IDhi19 corresponds to a Reserved Event event (0x4033) **0b0** The Common event is not implemented, or not counted.	x
[50]	IDhi18	IDhi18 corresponds to a Reserved Event event (0x4032) **0b0** The Common event is not implemented, or not counted.	x
[49]	IDhi17	IDhi17 corresponds to a Reserved Event event (0x4031) **0b0** The Common event is not implemented, or not counted.	x
[48]	IDhi16	IDhi16 corresponds to a Reserved Event event (0x4030) **0b0** The Common event is not implemented, or not counted.	x
[47]	IDhi15	IDhi15 corresponds to a Reserved Event event (0x402f) **0b0** The Common event is not implemented, or not counted.	x
[46]	IDhi14	IDhi14 corresponds to a Reserved Event event (0x402e) **0b0** The Common event is not implemented, or not counted.	x

Bits	Name	Description	Reset
[45]	IDhi13	IDhi13 corresponds to a Reserved Event event (0x402d) **0b0** The Common event is not implemented, or not counted.	x
[44]	IDhi12	IDhi12 corresponds to a Reserved Event event (0x402c) **0b0** The Common event is not implemented, or not counted.	x
[43]	IDhi11	IDhi11 corresponds to a Reserved Event event (0x402b) **0b0** The Common event is not implemented, or not counted.	x
[42]	IDhi10	IDhi10 corresponds to a Reserved Event event (0x402a) **0b0** The Common event is not implemented, or not counted.	x
[41]	IDhi9	IDhi9 corresponds to a Reserved Event event (`0x4029`) **0b0** The Common event is not implemented, or not counted.	x
[40]	IDhi8	IDhi8 corresponds to a Reserved Event event (`0x4028`) **0b0** The Common event is not implemented, or not counted.	x
[39]	IDhi7	IDhi7 corresponds to a Reserved Event event (`0x4027`) **0b0** The Common event is not implemented, or not counted.	x
[38]	IDhi6	IDhi6 corresponds to common event (`0x4026`) MEM_ACCESS_WR_CHECKED **0b1** The Common event is implemented.	x
[37]	IDhi5	IDhi5 corresponds to common event (`0x4025`) MEM_ACCESS_RD_CHECKED **0b1** The Common event is implemented.	x
[36]	IDhi4	IDhi4 corresponds to common event (`0x4024`) MEM_ACCESS_CHECKED **0b1** The Common event is implemented.	x
[35]	IDhi3	IDhi3 corresponds to common event (`0x4023`) Reserved **0b0** The Common event is not implemented, or not counted.	x
[34]	IDhi2	IDhi2 corresponds to common event (`0x4022`) ST_ALIGN_LAT **0b1** The Common event is implemented.	x
[33]	IDhi1	IDhi1 corresponds to common event (`0x4021`) LD_ALIGN_LAT **0b1** The Common event is implemented.	x
[32]	IDhi0	IDhi0 corresponds to common event (`0x4020`) LDST_ALIGN_LAT **0b1** The Common event is implemented.	x

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x3f) STALL_SLOT **0b1** The Common event is implemented.	x
[30]	ID30	ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND **0b1** The Common event is implemented.	x
[29]	ID29	ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND **0b1** The Common event is implemented.	x
[28]	ID28	ID28 corresponds to common event (0x3c) STALL **0b1** The Common event is implemented.	x
[27]	ID27	ID27 corresponds to common event (0x3b) OP_SPEC **0b1** The Common event is implemented.	x
[26]	ID26	ID26 corresponds to common event (0x3a) OP_RETIRED **0b1** The Common event is implemented.	x
[25]	ID25	ID25 corresponds to common event (`0x39`) L1D_CACHE_LMISS_RD **0b1** The Common event is implemented.	x
[24]	ID24	ID24 corresponds to common event (`0x38`) REMOTE_ACCESS_RD **0b1** The Common event is implemented.	x
[23]	ID23	ID23 corresponds to common event (`0x37`) LL_CACHE_MISS_RD **0b1** The Common event is implemented.	x
[22]	ID22	ID22 corresponds to common event (`0x36`) LL_CACHE_RD **0b1** The Common event is implemented.	x
[21]	ID21	ID21 corresponds to common event (`0x35`) ITLB_WLK **0b1** The Common event is implemented.	x
[20]	ID20	ID20 corresponds to common event (`0x34`) DTLB_WLK **0b1** The Common event is implemented.	x
[19]	ID19	ID19 corresponds to a Reserved Event event (`0x33`) **0b0** The Common event is not implemented, or not counted.	x
[18]	ID18	ID18 corresponds to a Reserved Event event (`0x32`) **0b0** The Common event is not implemented, or not counted.	x

Bits	Name	Description	Reset
[17]	ID17	ID17 corresponds to common event (0x31) REMOTE_ACCESS **0b0** The Common event is not implemented, or not counted.	x
[16]	ID16	ID16 corresponds to common event (0x30) L2I_TLB **0b0** The Common event is not implemented, or not counted.	x
[15]	ID15	ID15 corresponds to common event (0x2f) L2D_TLB **0b1** The Common event is implemented.	x
[14]	ID14	ID14 corresponds to common event (0x2e) L2I_TLB_REFILL **0b0** The Common event is not implemented, or not counted.	x
[13]	ID13	ID13 corresponds to common event (0x2d) L2D_TLB_REFILL **0b1** The Common event is implemented.	x
[12]	ID12	ID12 corresponds to common event (0x2c) Reserved **0b0** The Common event is not implemented, or not counted.	x
[11]	ID11	ID11 corresponds to common event (0x2b) L3D_CACHE **0b0** The common event is not implemented, or not counted. This value is reported if either the Cortex-A520 complex is configured without an L2 cache or the DSU is configured without an L3 cache. **0b1** The common event is implemented. This value is reported if both the Cortex-A520 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	x
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL **0b0** The Common event is not implemented, or not counted.	x
[9]	ID9	ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE **0b0** The Common event is not implemented, or not counted.	x
[8]	ID8	ID8 corresponds to common event (0x28) L2I_CACHE_REFILL **0b0** The Common event is not implemented, or not counted.	x
[7]	ID7	ID7 corresponds to common event (0x27) L2I_CACHE **0b0** The Common event is not implemented, or not counted.	x
[6]	ID6	ID6 corresponds to common event (0x26) L1I_TLB **0b1** The Common event is implemented.	x

Bits	Name	Description	Reset
[5]	ID5	ID5 corresponds to common event (0x25) L1D_TLB **0b1** The Common event is implemented.	x
[4]	ID4	ID4 corresponds to common event (0x24) STALL_BACKEND **0b1** The Common event is implemented.	x
[3]	ID3	ID3 corresponds to common event (0x23) STALL_FRONTEND **0b1** The Common event is implemented.	x
[2]	ID2	ID2 corresponds to common event (0x22) BR_MIS_PRED_RETIRED **0b1** The Common event is implemented.	x
[1]	ID1	ID1 corresponds to common event (0x21) BR_RETIRED **0b1** The Common event is implemented.	x
[0]	ID0	ID0 corresponds to common event (0x20) L2D_CACHE_ALLOCATE **0b0** The common event is not implemented, or not counted. This value is reported if the Cortex-A520 complex is configured without an L2 cache. **0b1** The common event is implemented. This value is reported if the Cortex-A520 complex is configured with an L2 cache.	x

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 Halted() && EDSCR.SDD == '1' && 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() && HCR_EL2.<E2H,TGE> != '11' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID1_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID1_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID1_EL0;
elsif PSTATE.EL == EL3 then
    return PMCEID1_EL0;
```

A.8 AArch64 Generic Timer registers summary

The summary table provides an overview of all Generic 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-224: Generic Timer registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
CNTKCTL_EL1	3	0	C14	C1	0	–	64-bit	Counter-timer Kernel Control register
CNTFRQ_EL0	3	3	C14	C0	0	–	64-bit	Counter-timer Frequency register
CNTPCT_EL0	3	3	C14	C0	1	–	64-bit	Counter-timer Physical Count register
CNTVCT_EL0	3	3	C14	C0	2	–	64-bit	Counter-timer Virtual Count register
CNTPCTSS_EL0	3	3	C14	C0	5	–	64-bit	Counter-timer Self-Synchronized Physical Count register
CNTVCTSS_EL0	3	3	C14	C0	6	–	64-bit	Counter-timer Self-Synchronized Virtual Count register
CNTP_TVAL_EL0	3	3	C14	C2	0	–	64-bit	Counter-timer Physical Timer TimerValue register
CNTP_CTL_EL0	3	3	C14	C2	1	–	64-bit	Counter-timer Physical Timer Control register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
CNTP_CVAL_EL0	3	3	C14	C2	2	—	64-bit	Counter-timer Physical Timer CompareValue register
CNTV_TVAL_EL0	3	3	C14	C3	0	—	64-bit	Counter-timer Virtual Timer TimerValue register
CNTV_CTL_EL0	3	3	C14	C3	1	—	64-bit	Counter-timer Virtual Timer Control register
CNTV_CVAL_EL0	3	3	C14	C3	2	—	64-bit	Counter-timer Virtual Timer CompareValue register
CNTVOFF_EL2	3	4	C14	C0	3	—	64-bit	Counter-timer Virtual Offset register
CNTPOFF_EL2	3	4	C14	C0	6	—	64-bit	Counter-timer Physical Offset register
CNTHCTL_EL2	3	4	C14	C1	0	—	64-bit	Counter-timer Hypervisor Control register
CNTHP_TVAL_EL2	3	4	C14	C2	0	—	64-bit	Counter-timer Physical Timer TimerValue register (EL2)
CNTHP_CTL_EL2	3	4	C14	C2	1	—	64-bit	Counter-timer Hypervisor Physical Timer Control register
CNTHP_CVAL_EL2	3	4	C14	C2	2	—	64-bit	Counter-timer Physical Timer CompareValue register (EL2)
CNTHV_TVAL_EL2	3	4	C14	C3	0	—	64-bit	Counter-timer Virtual Timer TimerValue Register (EL2)
CNTHV_CTL_EL2	3	4	C14	C3	1	—	64-bit	Counter-timer Virtual Timer Control register (EL2)
CNTHV_CVAL_EL2	3	4	C14	C3	2	—	64-bit	Counter-timer Virtual Timer CompareValue register (EL2)
CNTHVS_TVAL_EL2	3	4	C14	C4	0	—	64-bit	Counter-timer Secure Virtual Timer TimerValue register (EL2)
CNTHVS_CTL_EL2	3	4	C14	C4	1	—	64-bit	Counter-timer Secure Virtual Timer Control register (EL2)
CNTHVS_CVAL_EL2	3	4	C14	C4	2	—	64-bit	Counter-timer Secure Virtual Timer CompareValue register (EL2)
CNTHPS_TVAL_EL2	3	4	C14	C5	0	—	64-bit	Counter-timer Secure Physical Timer TimerValue register (EL2)
CNTHPS_CTL_EL2	3	4	C14	C5	1	—	64-bit	Counter-timer Secure Physical Timer Control register (EL2)
CNTHPS_CVAL_EL2	3	4	C14	C5	2	—	64-bit	Counter-timer Secure Physical Timer CompareValue register (EL2)
CNTPS_TVAL_EL1	3	7	C14	C2	0	—	64-bit	Counter-timer Physical Secure Timer TimerValue register
CNTPS_CTL_EL1	3	7	C14	C2	1	—	64-bit	Counter-timer Physical Secure Timer Control register
CNTPS_CVAL_EL1	3	7	C14	C2	2	—	64-bit	Counter-timer Physical Secure Timer CompareValue register

A.9 AArch64 Other system control registers summary

The summary table provides an overview of all Other 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-225: Other system control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SCTLR_EL1	3	0	C1	C0	0	—	64-bit	System Control Register (EL1)
CPACR_EL1	3	0	C1	C0	2	—	64-bit	Architectural Feature Access Control Register
ZCR_EL1	3	0	C1	C2	0	—	64-bit	SVE Control Register (EL1)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SCTLR_EL2	3	4	C1	C0	0	—	64-bit	System Control Register (EL2)
HCR_EL2	3	4	C1	C1	0	—	64-bit	Hypervisor Configuration Register
CPTR_EL2	3	4	C1	C1	2	—	64-bit	Architectural Feature Trap Register (EL2)
HSTR_EL2	3	4	C1	C1	3	—	64-bit	Hypervisor System Trap Register
HFGRTR_EL2	3	4	C1	C1	4	—	64-bit	Hypervisor Fine-Grained Read Trap Register
HFGWTR_EL2	3	4	C1	C1	5	—	64-bit	Hypervisor Fine-Grained Write Trap Register
HFGITR_EL2	3	4	C1	C1	6	—	64-bit	Hypervisor Fine-Grained Instruction Trap Register
ZCR_EL2	3	4	C1	C2	0	—	64-bit	SVE Control Register (EL2)
HCRX_EL2	3	4	C1	C2	2	—	64-bit	Extended Hypervisor Configuration Register
HDFGRTR_EL2	3	4	C3	C1	4	—	64-bit	Hypervisor Debug Fine-Grained Read Trap Register
HDFGWTR_EL2	3	4	C3	C1	5	—	64-bit	Hypervisor Debug Fine-Grained Write Trap Register
HAFGRTR_EL2	3	4	C3	C1	6	—	64-bit	Hypervisor Activity Monitors Fine-Grained Read Trap Register
SCTLR_EL3	3	6	C1	C0	0	—	64-bit	System Control Register (EL3)
ZCR_EL3	3	6	C1	C2	0	—	64-bit	SVE Control Register (EL3)

A.10 AArch64 Memory Partitioning and Monitoring registers summary

The summary table provides an overview of all Memory Partitioning and Monitoring 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-226: Memory Partitioning and Monitoring registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
MPAM1_EL1	3	0	C10	C5	0	—	64-bit	MPAM1 Register (EL1)
MPAM0_EL1	3	0	C10	C5	1	—	64-bit	MPAM0 Register (EL1)
MPAMHCR_EL2	3	4	C10	C4	0	—	64-bit	MPAM Hypervisor Control Register (EL2)
MPAMVPMV_EL2	3	4	C10	C4	1	—	64-bit	MPAM Virtual Partition Mapping Valid Register
MPAM2_EL2	3	4	C10	C5	0	—	64-bit	MPAM2 Register (EL2)
MPAMVPM0_EL2	3	4	C10	C6	0	—	64-bit	MPAM Virtual PARTID Mapping Register 0
MPAMVPM1_EL2	3	4	C10	C6	1	—	64-bit	MPAM Virtual PARTID Mapping Register 1
MPAM3_EL3	3	6	C10	C5	0	—	64-bit	MPAM3 Register (EL3)

A.11 AArch64 Activity Monitors registers summary

The summary table provides an overview of all Activity Monitors 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-227: Activity Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMCR_EL0	3	3	C13	C2	0	—	64-bit	Activity Monitors Control Register
AMCFGR_EL0	3	3	C13	C2	1	—	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	3	3	C13	C2	2	—	64-bit	Activity Monitors Counter Group Configuration Register
AMUSERENR_EL0	3	3	C13	C2	3	—	64-bit	Activity Monitors User Enable Register
AMCNTENCLR0_EL0	3	3	C13	C2	4	—	64-bit	Activity Monitors Count Enable Clear Register 0
AMCNTENSET0_EL0	3	3	C13	C2	5	—	64-bit	Activity Monitors Count Enable Set Register 0
AMCNTENCLR1_EL0	3	3	C13	C3	0	—	64-bit	Activity Monitors Count Enable Clear Register 1
AMCNTENSET1_EL0	3	3	C13	C3	1	—	64-bit	Activity Monitors Count Enable Set Register 1
AMEVCNTR00_EL0	3	3	C13	C4	0	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR01_EL0	3	3	C13	C4	1	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR02_EL0	3	3	C13	C4	2	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR03_EL0	3	3	C13	C4	3	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVTYPER00_EL0	3	3	C13	C6	0	—	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	3	C13	C6	1	—	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	3	C13	C6	2	—	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	3	C13	C6	3	—	64-bit	Activity Monitors Event Type Registers 0
AMEVCNTR10_EL0	3	3	C13	C12	0	—	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR11_EL0	3	3	C13	C12	1	—	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR12_EL0	3	3	C13	C12	2	—	64-bit	Activity Monitors Event Counter Registers 1
AMEVTYPER10_EL0	3	3	C13	C14	0	—	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	3	C13	C14	1	—	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	3	C13	C14	2	—	64-bit	Activity Monitors Event Type Registers 1

A.11.1 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 System register B.6.9 AMCFGR, Activity Monitors Configuration Register on page 695 bits [31:0].

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0001 xxx1 0000 0000 0011 1111 0000 0110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-89: AArch64_amcfgr_el0 bit assignments

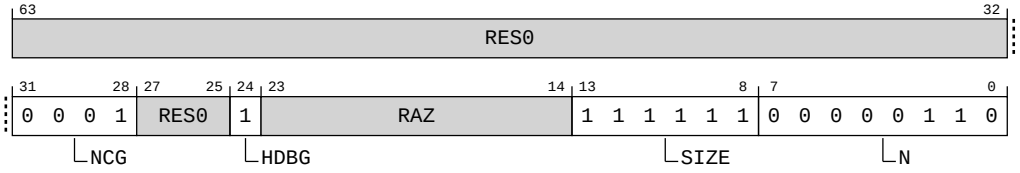


Table A-228: AMCFGR_EL0 bit descriptions

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

Bits	Name	Description	Reset
[27:25]	**RES0**	Reserved	**RES0**
[24]	HDBG	Halt-on-debug supported. This feature must be supported, and so this bit is `0b1`. **0b1** AArch64-AMCR_EL0.HDBG is read/write.	`0b1`
[23:14]	**RAZ**	Reserved	**RAZ**
[13:8]	SIZE	Defines the size of activity monitor event counters. The size of the activity monitor event counters implemented by the activity monitors Extension is [AMCFGR_EL0.SIZE + 1]. **Note:** Software also uses this field to determine the spacing of counters in the memory-map. From Armv8, the counters are at doubleword-aligned addresses. **0b111111**	`0b111111`
[7:0]	N	Defines the number of activity monitor event counters. The total number of counters implemented in all groups by the Activity Monitors Extension is [AMCFGR_EL0.N + 1]. **0b00000110** Seven activity monitor event counters	`0x06`

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCFGR_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
```

```
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCFGR_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCFGR_EL0;
elsif PSTATE.EL == EL3 then
    return AMCFGR_EL0;
```

A.11.2 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 System register B.6.8 AMCGCR, Activity Monitors Counter Group Configuration Register on page 693 bits [31:0].

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0011 0000 0100

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-90: AArch64_amcgcr_el0 bit assignments

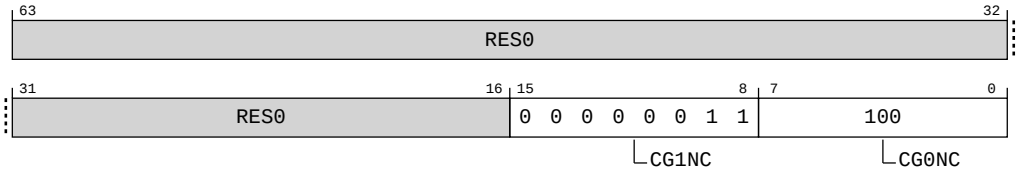


Table A-230: AMCGCR_EL0 bit descriptions

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. In an implementation that includes FEAT_AMUv1, the permitted range of values is 0x0 to 0x10. **0b00000011** Three counters in the auxiliary counter group	0x03
[7:0]	CG0NC	Counter Group 0 Number of Counters. The number of counters in the architected counter group. **0b00000100**	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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCGCR_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
```

```
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCGCR_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCGCR_EL0;
elsif PSTATE.EL == EL3 then
    return AMCGCR_EL0;
```

A.11.3 AMEVTYPER00_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR00_EL0 counts.

Configurations

AArch64 register AMEVTYPER00_EL0 bits [31:0] are architecturally mapped to External System register B.6.1 AMEVTYPER00, Activity Monitors Event Type Registers 0 on page 681 bits [31:0].

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Figure A-91: AArch64_amevtyper00_el0 bit assignments

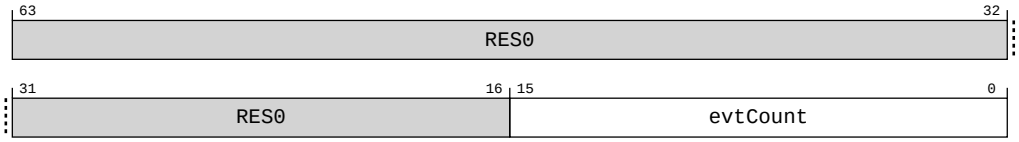


Table A-232: AMEVTYPER00_EL0 bit descriptions

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 AArch64-AMEVCNTR0<n>_EL0. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0000000000010001** Processor frequency cycles	16{x}

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER00_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b000

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are UNDEFINED.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER00_EL0

```
if PSTATE.EL == EL0 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER00_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER00_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER00_EL0;
elsif PSTATE.EL == EL3 then
    return AMEVTYPER00_EL0;
```

A.11.4 AMEVTYPER01_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR01_EL0 counts.

Configurations

AArch64 register AMEVTYPER01_EL0 bits [31:0] are architecturally mapped to External System register B.6.2 AMEVTYPER01, Activity Monitors Event Type Registers 0 on page 683 bits [31:0].

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Figure A-92: AArch64_amevtyper01_el0 bit assignments

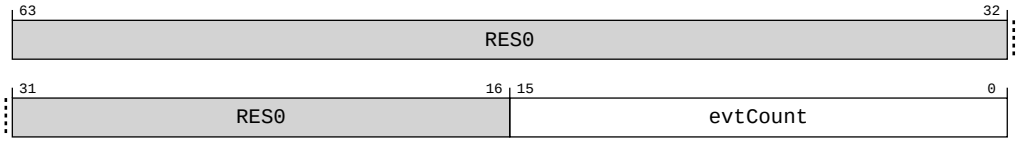


Table A-234: AMEVTYPER01_EL0 bit descriptions

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 AArch64-AMEVCNTR0<n>_EL0. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0100000000000100** Constant frequency cycles	16{x}

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER01_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b001

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are UNDEFINED.


Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER01_EL0

```
if PSTATE.EL == EL0 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER01_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER01_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER01_EL0;
elsif PSTATE.EL == EL3 then
    return AMEVTYPER01_EL0;
```

A.11.5 AMEVTYPER02_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR02_EL0 counts.

Configurations

AArch64 register AMEVTYPER02_EL0 bits [31:0] are architecturally mapped to External System register B.6.3 AMEVTYPER02, Activity Monitors Event Type Registers 0 on page 685 bits [31:0].

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Figure A-93: AArch64_amevtyper02_el0 bit assignments

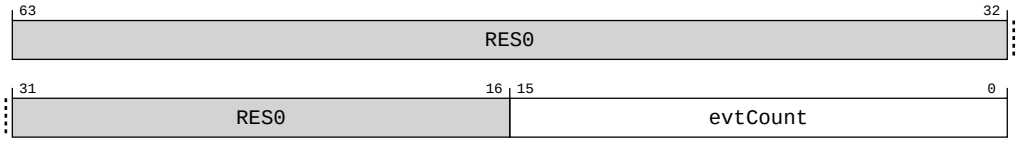


Table A-236: AMEVTYPER02_EL0 bit descriptions

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 AArch64-AMEVCNTR0<n>_EL0. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0000000000001000** Instructions retired	16{x}

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are **UNDEFINED**.

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER02_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b010

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are UNDEFINED.

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER02_EL0

```
if PSTATE.EL == EL0 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER02_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER02_EL0;
elsif PSTATE.EL == EL2 then
```

```
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER02_EL0;
elsif PSTATE.EL == EL3 then
    return AMEVTYPER02_EL0;
```

A.11.6 AMEVTYPER03_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR03_EL0 counts.

Configurations

AArch64 register AMEVTYPER03_EL0 bits [31:0] are architecturally mapped to External System register B.6.4 AMEVTYPER03, Activity Monitors Event Type Registers 0 on page 686 bits [31:0].

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Figure A-94: AArch64_amevtyper03_el0 bit assignments

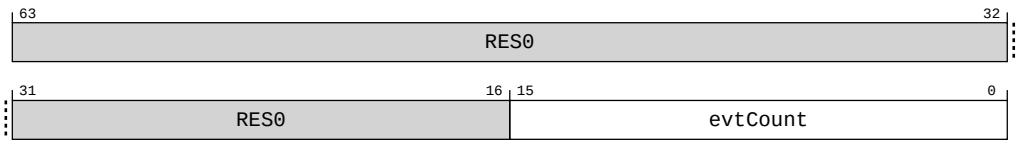


Table A-238: AMEVTYPER03_EL0 bit descriptions

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 AArch64-AMEVCNTR0<n>_EL0. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0100000000000101** Memory stall cycles	16{x}

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are **UNDEFINED**.

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER03_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b011

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are UNDEFINED.

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER03_EL0

```
if PSTATE.EL == EL0 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER03_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER03_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER03_EL0;
elsif PSTATE.EL == EL3 then
    return AMEVTYPER03_EL0;
```

A.11.7 AMEVTYPER10_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR10_EL0 counts.

Configurations

AArch64 register AMEVTYPER10_EL0 bits [31:0] are architecturally mapped to External System register B.6.5 AMEVTYPER10, Activity Monitors Event Type Registers 1 on page 688 bits [31:0].

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-95: AArch64_amevtyper10_el0 bit assignments

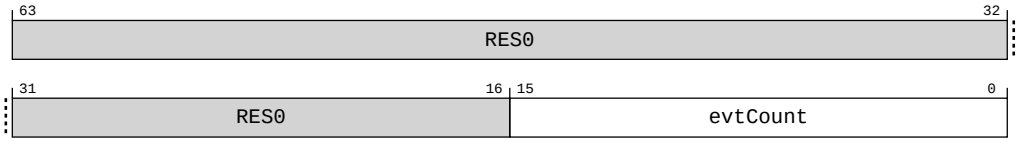


Table A-240: AMEVTYPER10_EL0 bit descriptions

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 AMEVCNTR10_EL0. **0b0000001100000000** MPMM gear 0 period threshold exceeded	`16{x}`

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER10_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b000

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are UNDEFINED.

Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER10_EL0

```
if PSTATE.EL == EL0 then
    if Halted() && EDSCR.SDD == '1' && 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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER10_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER10_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER10_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER10_EL0;
elsif PSTATE.EL == EL3 then
    return AMEVTYPER10_EL0;
```

A.11.8 AMEVTYPER11_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR11_EL0 counts.

Configurations

AArch64 register AMEVTYPER11_EL0 bits [31:0] are architecturally mapped to External System register B.6.6 AMEVTYPER11, Activity Monitors Event Type Registers 1 on page 690 bits [31:0].

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Figure A-96: AArch64_amevtyper11_el0 bit assignments

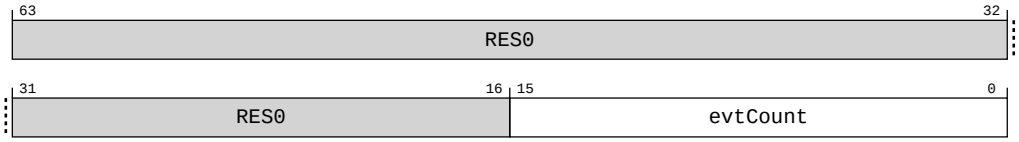


Table A-242: AMEVTYPER11_EL0 bit descriptions

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. **0b0000001100000001** MPMM gear 1 period threshold exceeded	16{x}

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER11_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b001

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are UNDEFINED.


Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER11_EL0

```
if PSTATE.EL == EL0 then
    if Halted() && EDSCR.SDD == '1' && 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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER11_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER11_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER11_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER11_EL0;
elsif PSTATE.EL == EL3 then
    return AMEVTYPER11_EL0;
```

A.11.9 AMEVTYPER12_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR12_EL0 counts.

Configurations

AArch64 register AMEVTYPER12_EL0 bits [31:0] are architecturally mapped to External System register B.6.7 AMEVTYPER12, Activity Monitors Event Type Registers 1 on page 692 bits [31:0].

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Figure A-97: AArch64_amevtyper12_el0 bit assignments

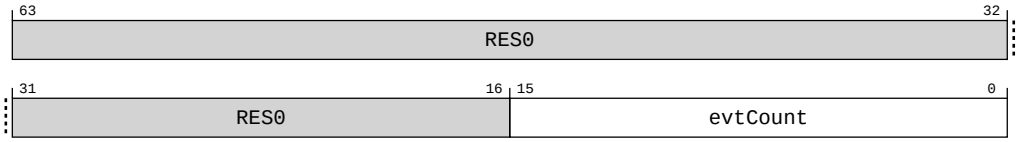


Table A-244: AMEVTYPER12_EL0 bit descriptions

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. **0b0000001100000010** MPMM gear 2 period threshold exceeded	16{x}

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are **UNDEFINED**.

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER12_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b010

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are UNDEFINED.

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER12_EL0

```
if PSTATE.EL == EL0 then
    if Halted() && EDSCR.SDD == '1' && 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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER12_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER12_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER12_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMEVTYPER12_EL0;
elsif PSTATE.EL == EL3 then
    return AMEVTYPER12_EL0;
```

A.12 AArch64 RAS registers summary

The 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-246: RAS registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ERRIDR_EL1	3	0	C5	C3	0	–	64-bit	Error Record ID Register
ERRSELR_EL1	3	0	C5	C3	1	–	64-bit	Error Record Select Register
ERXFR_EL1	3	0	C5	C4	0	–	64-bit	Selected Error Record Feature Register
ERXCTLR_EL1	3	0	C5	C4	1	–	64-bit	Selected Error Record Control Register
ERXSTATUS_EL1	3	0	C5	C4	2	–	64-bit	Selected Error Record Primary Status Register
ERXADDR_EL1	3	0	C5	C4	3	–	64-bit	Selected Error Record Address Register
ERXPFGF_EL1	3	0	C5	C4	4	–	64-bit	Selected Pseudo-fault Generation Feature register
ERXPFGCTL_EL1	3	0	C5	C4	5	–	64-bit	Selected Pseudo-fault Generation Control register
ERXPFGCDN_EL1	3	0	C5	C4	6	–	64-bit	Selected Pseudo-fault Generation Countdown register
ERXMISC0_EL1	3	0	C5	C5	0	–	64-bit	Selected Error Record Miscellaneous Register 0
ERXMISC1_EL1	3	0	C5	C5	1	–	64-bit	Selected Error Record Miscellaneous Register 1
ERXMISC2_EL1	3	0	C5	C5	2	–	64-bit	Selected Error Record Miscellaneous Register 2
ERXMISC3_EL1	3	0	C5	C5	3	–	64-bit	Selected Error Record Miscellaneous Register 3
DISR_EL1	3	0	C12	C1	1	–	64-bit	Deferred Interrupt Status Register
VSESR_EL2	3	4	C5	C2	3	–	64-bit	Virtual SError Exception Syndrome Register
VDISR_EL2	3	4	C12	C1	1	–	64-bit	Virtual Deferred Interrupt Status Register

A.12.1 ERRIDR_EL1, Error Record ID Register

Defines the highest numbered index of the error records that can be accessed through the Error Record System registers.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

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

Figure A-98: AArch64_erridr_el1 bit assignments

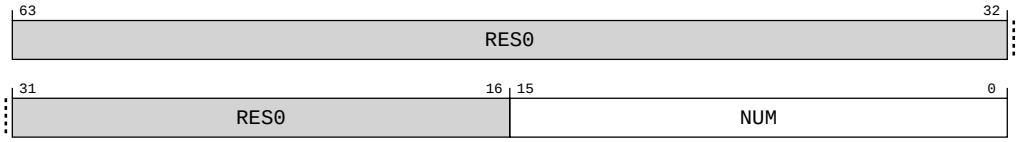


Table A-247: ERRIDR_EL1 bit descriptions

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. **0b0000000000000011** Three Records Present.	16{x}

Access

MRS <Xt>, ERRIDR_EL1

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

Accessibility

MRS <Xt>, ERRIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRIDR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRIDR_EL1;
elsif PSTATE.EL == EL3 then
    return ERRIDR_EL1;
```

A.12.2 ERRSELR_EL1, Error Record Select Register

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

Configurations

If AArch64-ERRIDR_EL1 indicates that zero error records are implemented, then it is IMPLEMENTATION DEFINED whether ERRSELR_EL1 is UNDEFINED or RES0.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

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

Figure A-99: AArch64_errselr_el1 bit assignments

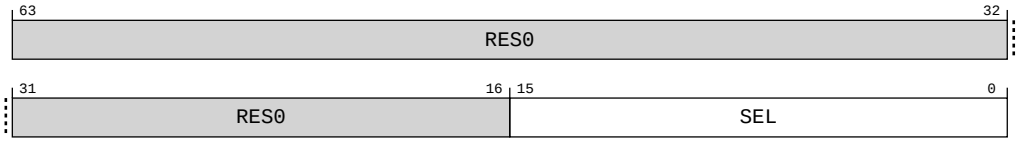


Table A-249: ERRSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15:0]	SEL	Selects the error record accessed through the ERX registers. **0b0000000000000000** Selects record 0, containing errors from DSU RAMs **0b0000000000000001** Selects record 1, containing errors from L1 RAMs **0b0000000000000010** Selects record 2, containing errors from L2 RAMs	16{x}

Access

MRS <Xt>, ERRSELR_EL1

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

MSR ERRSELR_EL1, <Xt>

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

Accessibility

MRS <Xt>, ERRSELR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRSELR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRSELR_EL1;
elsif PSTATE.EL == EL3 then
    return ERRSELR_EL1;
```

MSR ERRSELR_EL1, <Xt>

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

A.13 AArch64 Trace unit registers summary

The summary table provides an overview of all Trace unit 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-252: Trace unit registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCTRACEIDR	2	1	C0	C0	1	–	64-bit	Trace ID Register
TRCVICTLR	2	1	C0	C0	2	–	64-bit	ViewInst Main Control Register
TRCSEQEVR0	2	1	C0	C0	4	–	64-bit	Sequencer State Transition Control Register <n>
TRCCNTRLDVR0	2	1	C0	C0	5	–	64-bit	Counter Reload Value Register <n>
TRCIDR8	2	1	C0	C0	6	–	64-bit	ID Register 8
TRCIMSPEC0	2	1	C0	C0	7	–	64-bit	IMP DEF Register 0
TRCPRGCTLR	2	1	C0	C1	0	–	64-bit	Programming Control Register
TRCVIIECTLR	2	1	C0	C1	2	–	64-bit	ViewInst Include/Exclude Control Register
TRCSEQEVR1	2	1	C0	C1	4	–	64-bit	Sequencer State Transition Control Register <n>
TRCCNTRLDVR1	2	1	C0	C1	5	–	64-bit	Counter Reload Value Register <n>
TRCIDR9	2	1	C0	C1	6	–	64-bit	ID Register 9
TRCVISSCTLR	2	1	C0	C2	2	–	64-bit	ViewInst Start/Stop Control Register
TRCSEQEVR2	2	1	C0	C2	4	–	64-bit	Sequencer State Transition Control Register <n>
TRCIDR10	2	1	C0	C2	6	–	64-bit	ID Register 10
TRCSTATR	2	1	C0	C3	0	–	64-bit	Trace Status Register
TRCIDR11	2	1	C0	C3	6	–	64-bit	ID Register 11
TRCCONFIGR	2	1	C0	C4	0	–	64-bit	Trace Configuration Register
TRCCNTCTLR0	2	1	C0	C4	5	–	64-bit	Counter Control Register <n>
TRCIDR12	2	1	C0	C4	6	–	64-bit	ID Register 12
TRCCNTCTLR1	2	1	C0	C5	5	–	64-bit	Counter Control Register <n>
TRCIDR13	2	1	C0	C5	6	–	64-bit	ID Register 13
TRCAUXCTLR	2	1	C0	C6	0	–	64-bit	Auxiliary Control Register
TRCSEQRSTEVR	2	1	C0	C6	4	–	64-bit	Sequencer Reset Control Register
TRCSEQSTR	2	1	C0	C7	4	–	64-bit	Sequencer State Register
TRCEVENTCTL0R	2	1	C0	C8	0	–	64-bit	Event Control 0 Register
TRCEXTINSELR0	2	1	C0	C8	4	–	64-bit	External Input Select Register <n>
TRCCNTVR0	2	1	C0	C8	5	–	64-bit	Counter Value Register <n>
TRCIDR0	2	1	C0	C8	7	–	64-bit	ID Register 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCEVENTCTL1R	2	1	C0	C9	0	–	64-bit	Event Control 1 Register
TRCEXTINSELR1	2	1	C0	C9	4	–	64-bit	External Input Select Register <n>
TRCCNTVR1	2	1	C0	C9	5	–	64-bit	Counter Value Register <n>
TRCIDR1	2	1	C0	C9	7	–	64-bit	ID Register 1
TRCRSR	2	1	C0	C10	0	–	64-bit	Resources Status Register
TRCEXTINSELR2	2	1	C0	C10	4	–	64-bit	External Input Select Register <n>
TRCIDR2	2	1	C0	C10	7	–	64-bit	ID Register 2
TRCSTALLCTLR	2	1	C0	C11	0	–	64-bit	Stall Control Register
TRCEXTINSELR3	2	1	C0	C11	4	–	64-bit	External Input Select Register <n>
TRCIDR3	2	1	C0	C11	7	–	64-bit	ID Register 3
TRCTSCTLR	2	1	C0	C12	0	–	64-bit	Timestamp Control Register
TRCIDR4	2	1	C0	C12	7	–	64-bit	ID Register 4
TRCSYNCPR	2	1	C0	C13	0	–	64-bit	Synchronization Period Register
TRCIDR5	2	1	C0	C13	7	–	64-bit	ID Register 5
TRCCCCTLR	2	1	C0	C14	0	–	64-bit	Cycle Count Control Register
TRCIDR6	2	1	C0	C14	7	–	64-bit	ID Register 6
TRCBBCTLR	2	1	C0	C15	0	–	64-bit	Branch Broadcast Control Register
TRCIDR7	2	1	C0	C15	7	–	64-bit	ID Register 7
TRCSSCCR0	2	1	C1	C0	2	–	64-bit	Single-shot Comparator Control Register <n>
TRCOSLSR	2	1	C1	C1	4	–	64-bit	Trace OS Lock Status Register
TRCRSCTLR2	2	1	C1	C2	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR3	2	1	C1	C3	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR4	2	1	C1	C4	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR5	2	1	C1	C5	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR6	2	1	C1	C6	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR7	2	1	C1	C7	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR8	2	1	C1	C8	0	–	64-bit	Resource Selection Control Register <n>
TRCSSCSR0	2	1	C1	C8	2	–	64-bit	Single-shot Comparator Control Status Register <n>
TRCRSCTLR9	2	1	C1	C9	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR10	2	1	C1	C10	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR11	2	1	C1	C11	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR12	2	1	C1	C12	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR13	2	1	C1	C13	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR14	2	1	C1	C14	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR15	2	1	C1	C15	0	–	64-bit	Resource Selection Control Register <n>
TRCACVR0	2	1	C2	C0	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR0	2	1	C2	C0	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR1	2	1	C2	C2	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR1	2	1	C2	C2	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR2	2	1	C2	C4	0	–	64-bit	Address Comparator Value Register <n>

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCACATR2	2	1	C2	C4	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR3	2	1	C2	C6	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR3	2	1	C2	C6	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR4	2	1	C2	C8	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR4	2	1	C2	C8	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR5	2	1	C2	C10	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR5	2	1	C2	C10	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR6	2	1	C2	C12	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR6	2	1	C2	C12	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR7	2	1	C2	C14	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR7	2	1	C2	C14	2	–	64-bit	Address Comparator Access Type Register <n>
TRCCIDCVR0	2	1	C3	C0	0	–	64-bit	Context Identifier Comparator Value Registers <n>
TRCVMIDCVR0	2	1	C3	C0	1	–	64-bit	Virtual Context Identifier Comparator Value Register <n>
TRCCIDCCTLR0	2	1	C3	C0	2	–	64-bit	Context Identifier Comparator Control Register 0
TRCVMIDCCTLR0	2	1	C3	C2	2	–	64-bit	Virtual Context Identifier Comparator Control Register 0
TRCDEVID	2	1	C7	C2	7	–	64-bit	Device Configuration Register
TRCCLAIMSET	2	1	C7	C8	6	–	64-bit	Claim Tag Set Register
TRCCLAIMCLR	2	1	C7	C9	6	–	64-bit	Claim Tag Clear Register
TRCAUTHSTATUS	2	1	C7	C14	6	–	64-bit	Authentication Status Register
TRCDEVARCH	2	1	C7	C15	6	–	64-bit	Device Architecture Register

A.13.1 TRCIDR8, 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 System register B.7.2 TRCIDR8, ID Register 8 on page 716 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-100: AArch64_trcidr8 bit assignments

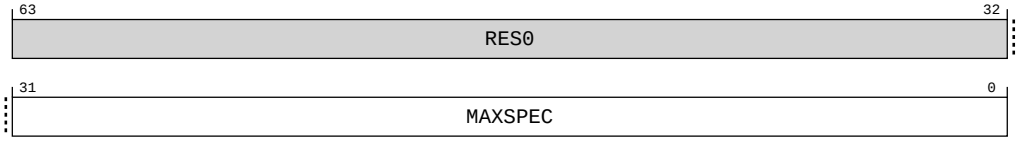


Table A-253: TRCIDR8 bit descriptions

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. **0b00000000000000000000000000000000**	32{x}

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR8;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
```

```
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR8;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR8;
```

A.13.2 TRCIMSPEC0, IMP DEF Register 0

TRCIMSPEC0 shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

Configurations

AArch64 register TRCIMSPEC0 bits [31:0] are architecturally mapped to External System register B.7.8 TRCIMSPEC0, IMP DEF Register 0 on page 723 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

RO

Reset value

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

Figure A-101: AArch64_trcimspec0 bit assignments

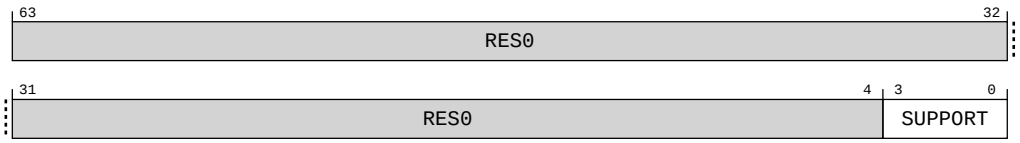


Table A-255: TRCIMSPEC0 bit descriptions

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.	xxxx

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPEC0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPEC0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPEC0;
```

MSR TRCIMSPEC0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t];
```

A.13.3 TRCIDR9, ID Register 9

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR9 bits [31:0] are architecturally mapped to External System register B.7.3 TRCIDR9, ID Register 9 on page 717 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-102: AArch64_trcidr9 bit assignments

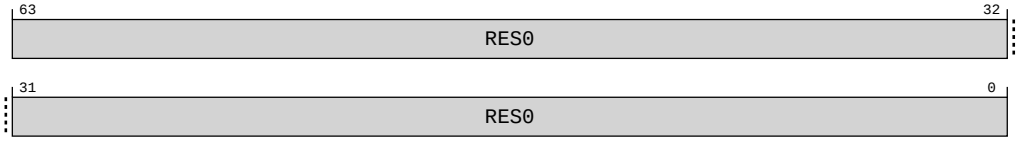


Table A-258: TRCIDR9 bit descriptions

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

Access

MRS <Xt>, TRCIDR9

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b110

Accessibility

MRS <Xt>, TRCIDR9

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR9;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR9;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR9;
```

A.13.4 TRCIDR10, ID Register 10

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR10 bits [31:0] are architecturally mapped to External System register B.7.4 TRCIDR10, ID Register 10 on page 719 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-103: AArch64_trcidr10 bit assignments

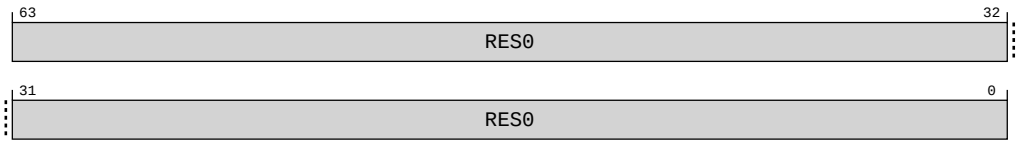


Table A-260: TRCIDR10 bit descriptions

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
```

A.13.5 TRCIDR11, ID Register 11

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR11 bits [31:0] are architecturally mapped to External System register B.7.5 TRCIDR11, ID Register 11 on page 720 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-104: AArch64_trcidr11 bit assignments

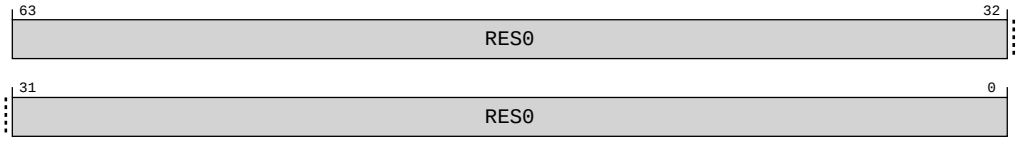


Table A-262: TRCIDR11 bit descriptions

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR11;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR11;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR11;
```

A.13.6 TRCIDR12, ID Register 12

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR12 bits [31:0] are architecturally mapped to External System register B.7.6 TRCIDR12, ID Register 12 on page 721 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-105: AArch64_trcidr12 bit assignments

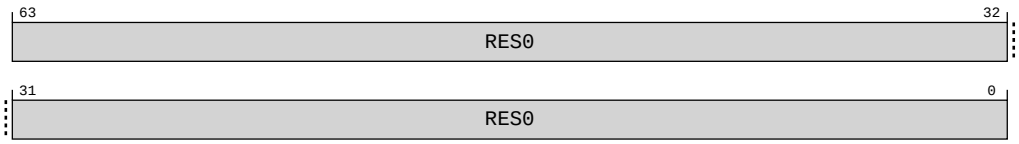


Table A-264: TRCIDR12 bit descriptions

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR12;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR12;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        return TRCIDR12;
```

A.13.7 TRCIDR13, ID Register 13

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR13 bits [31:0] are architecturally mapped to External System register B.7.7 TRCIDR13, ID Register 13 on page 722 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-106: AArch64_trcidr13 bit assignments

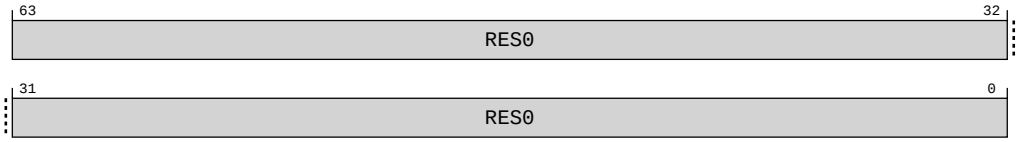


Table A-266: TRCIDR13 bit descriptions

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
```

A.13.8 TRCAUXCTLR, Auxiliary Control Register

The function of this register is **IMPLEMENTATION DEFINED**.

Configurations

AArch64 register TRCAUXCTLR bits [31:0] are architecturally mapped to External System register B.7.1 TRCAUXCTLR, Auxiliary Control Register on page 715 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-107: AArch64_trcauxctlr bit assignments

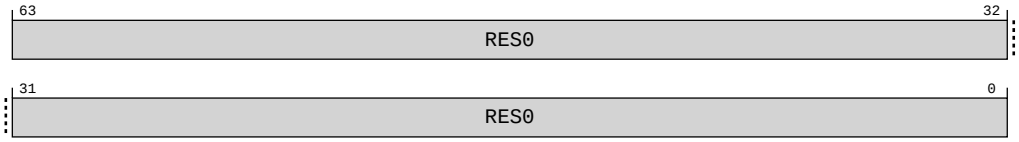


Table A-268: TRCAUXCTLR bit descriptions

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

Access

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.

MRS <Xt>, TRCAUXCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0110	0b000

MSR TRCAUXCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0110	0b000

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.

MRS <Xt>, TRCAUXCTLR

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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.TRCAUXCTLR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCAUXCTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCAUXCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCAUXCTLR;
```

MSR TRCAUXCTLR, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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.TRCAUXCTLR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCAUXCTLR = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCAUXCTLR = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCAUXCTLR = X[t];
```

A.13.9 TRCIDR0, ID Register 0

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR0 bits [31:0] are architecturally mapped to External System register B.7.9 TRCIDR0, ID Register 0 on page 724 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-108: AArch64_trcidr0 bit assignments

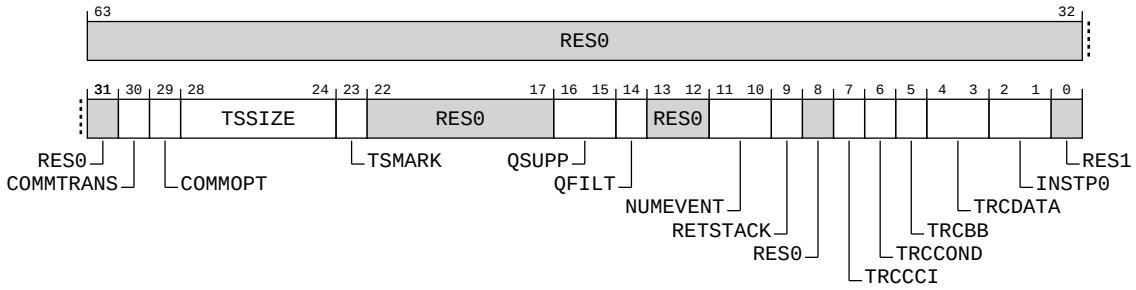


Table A-271: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30]	COMMTRANS	Transaction Start element behavior. **0b0** Transaction Start elements are P0 elements.	x
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **0b1** Commit mode 1. **When AArch64-TRCIDR8.MAXSPEC == 0x0** Access to this field is: **RAO/WI** **Otherwise** Access to this field is: RO	x
[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.	5{x}
[23]	TSMARK	Indicates whether Timestamp Marker elements are generated. **0b1** Timestamp Marker elements are generated.	x
[22:17]	**RES0**	Reserved	**RES0**
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **0b00** Q element support is not implemented.	xx
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	x
[13:12]	**RES0**	Reserved	**RES0**
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **0b11** The trace unit supports 4 ETEEvents.	xx
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **0b1** Return stack implemented.	x
[8]	**RES0**	Reserved	**RES0**
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **0b1** Cycle counting implemented.	x
[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.	x

Bits	Name	Description	Reset
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. **0b1** Branch broadcasting implemented.	x
[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.	xx
[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.	xx
[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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR0;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        return TRCIDR0;
```

A.13.10 TRCIDR1, ID Register 1

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR1 bits [31:0] are architecturally mapped to External System register B.7.10 TRCIDR1, ID Register 1 on page 727 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-109: AArch64_trcidr1 bit assignments

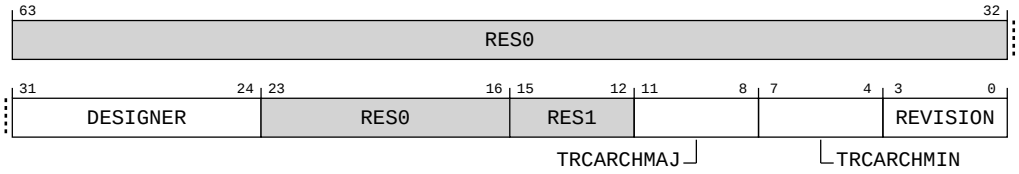


Table A-273: TRCIDR1 bit descriptions

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 AArch64-MIDR_EL1.Implementer. **0b01000000001** Arm Limited	8{x}

Bits	Name	Description	Reset
[23:16]	**RES0**	Reserved	**RES0**
[15:12]	**RES1**	Reserved	**RES1**
[11:8]	TRCARCHMAJ	Major architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH.	xxxx
[7:4]	TRCARCHMIN	Minor architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH.	xxxx
[3:0]	REVISION	Arm deprecates any use of this field. **0b0000** r0p2	xxxx

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        return TRCIDR1;
```

A.13.11 TRCIDR2, ID Register 2

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR2 bits [31:0] are architecturally mapped to External System register B.7.11 TRCIDR2, ID Register 2 on page 728 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-110: AArch64_trcidr2 bit assignments

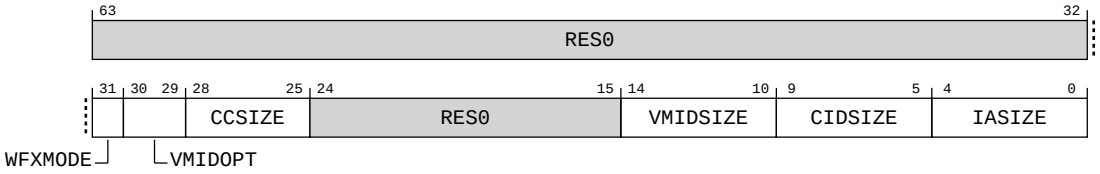


Table A-275: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	WFXMODE	Indicates whether WFI and WFE instructions are classified as P0 instructions: **0b1** WFI and WFE instructions are classified as P0 instructions.	x

Bits	Name	Description	Reset
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. **0b10** Virtual context identifier selection not supported. AArch64-TRCCONFIGR.VMIDOPT is **RES1**.	xx
[28:25]	CCSIZE	Indicates the size of the cycle counter. **0b0000** The cycle counter is 12 bits in length.	xxxx
[24:15]	**RES0**	Reserved	**RES0**
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. **0b00100** 32-bit Virtual context identifier size.	5{x}
[9:5]	CIDSIZE	Indicates the Context identifier size. **0b00100** 32-bit Context identifier size.	5{x}
[4:0]	IASIZE	Virtual instruction address size. **0b01000** Maximum of 64-bit instruction address size.	5{x}

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR2;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR2;
```

A.13.12 TRCIDR3, ID Register 3

Returns the base architecture of the trace unit.

Configurations

AArch64 register TRCIDR3 bits [31:0] are architecturally mapped to External System register B.7.12 TRCIDR3, ID Register 3 on page 730 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-111: AArch64_trcidr3 bit assignments

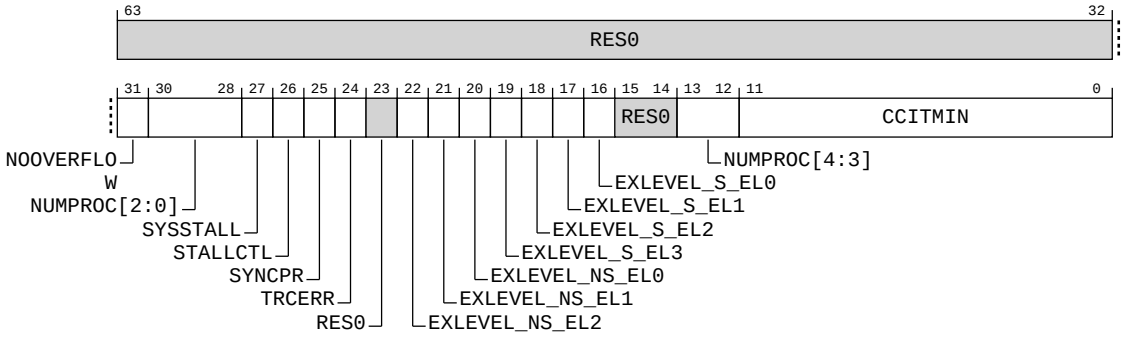


Table A-277: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	x
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **0b1** Stalling of the PE is permitted.	x
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **0b1** Stalling of the PE is implemented.	x
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0b0** AArch64-TRCSYNCPR is read-write so software can change the synchronization period.	x
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **0b1** Forced tracing of System Error exceptions is implemented.	x
[23]	**RES0**	Reserved	**RES0**
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 is implemented. **0b1** Non-secure EL2 is implemented.	x
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 is implemented. **0b1** Non-secure EL1 is implemented.	x
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 is implemented. **0b1** Non-secure EL0 is implemented.	x

Bits	Name	Description	Reset
[19]	EXLEVEL_S_EL3	Indicates if EL3 is implemented. **0b1** EL3 is implemented.	x
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 is implemented. **0b1** Secure EL2 is implemented.	x
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 is implemented. **0b1** Secure EL1 is implemented.	x
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 is implemented. **0b1** Secure EL0 is implemented.	x
[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.	5{x}
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in AArch64-TRCCCCTLR.THRESHOLD. If AArch64-TRCIDR0.TRCCCI == 1 then the minimum value of this field is 0x001. If AArch64-TRCIDR0.TRCCCI == 0 then this field is zero. **0b000000000100**	12{x}

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        return TRCIDR3;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR3;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR3;
```

A.13.13 TRCIDR4, ID Register 4

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR4 bits [31:0] are architecturally mapped to External System register B.7.13 TRCIDR4, ID Register 4 on page 732 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-112: AArch64_trcidr4 bit assignments

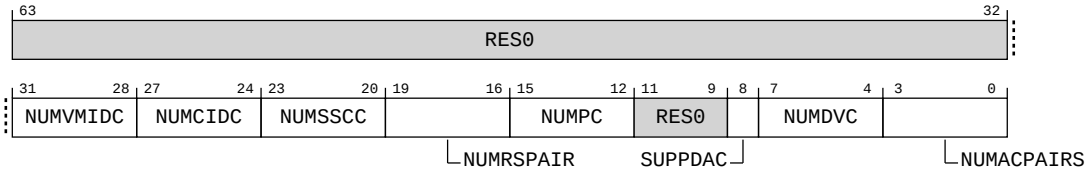


Table A-279: TRCIDR4 bit descriptions

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 implementation has one Virtual Context Identifier Comparator.	xxxx
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Context Identifier Comparator.	xxxx
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. **0b0001** The implementation has one Single-shot Comparator Control.	xxxx
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0b0111** The implementation has eight resource selector pairs.	xxxx
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0b0000** No PE Comparator Inputs are available.	xxxx
[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.	x
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0000** No data value comparators implemented.	xxxx
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **0b0100** The implementation has four Address Comparator pairs.	xxxx

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;
```

A.13.14 TRCIDR5, ID Register 5

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR5 bits [31:0] are architecturally mapped to External System register B.7.14 TRCIDR5, ID Register 5 on page 734 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-113: AArch64_trcidr5 bit assignments

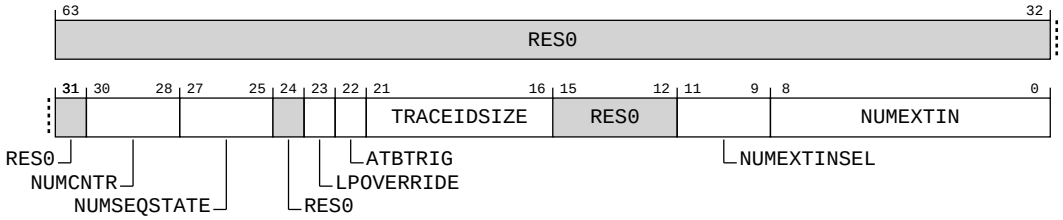


Table A-281: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. **0b010** Two Counters implemented.	xxx
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **0b100** Four Sequencer states are implemented.	xxx
[24]	**RES0**	Reserved	**RES0**
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **0b1** The trace unit supports Low-power Override Mode.	x
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **0b1** The implementation supports ATB triggers.	x
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **0b000111** The implementation supports a 7-bit trace ID.	6{x}
[15:12]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. **0b100** 4 External Input Selector resources are available.	xxx
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **0b111111111** Unified PMU event selection.	9{x}

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 Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;
```

A.13.15 TRCIDR6, ID Register 6

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR6 bits [31:0] are architecturally mapped to External System register B.7.15 TRCIDR6, ID Register 6 on page 736 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-114: AArch64_trcidr6 bit assignments

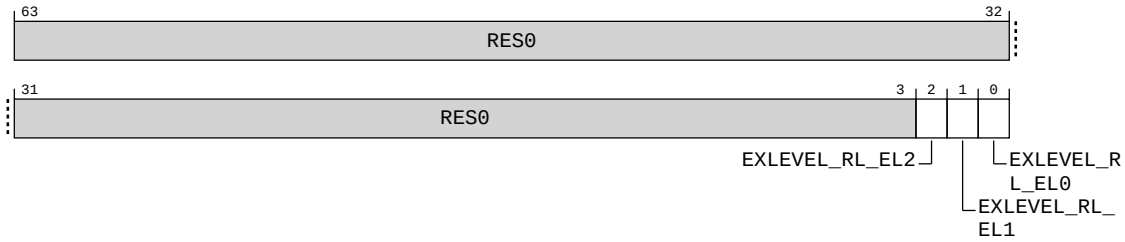


Table A-283: TRCIDR6 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0
[2]	EXLEVEL_RL_EL2	Indicates if Realm EL2 is implemented. **0b0** Realm EL2 is not implemented.	x

Bits	Name	Description	Reset
[1]	EXLEVEL_RL_EL1	Indicates if Realm EL1 is implemented. **0b0** Realm EL1 is not implemented.	x
[0]	EXLEVEL_RL_EL0	Indicates if Realm EL0 is implemented. **0b0** Realm EL0 is not implemented.	x

Access

MRS <Xt>, TRCIDR6

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1110	0b111

Accessibility

MRS <Xt>, TRCIDR6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR6;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR6;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR6;
```

A.13.16 TRCIDR7, ID Register 7

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR7 bits [31:0] are architecturally mapped to External System register B.7.16 TRCIDR7, ID Register 7 on page 738 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-115: AArch64_trcidr7 bit assignments

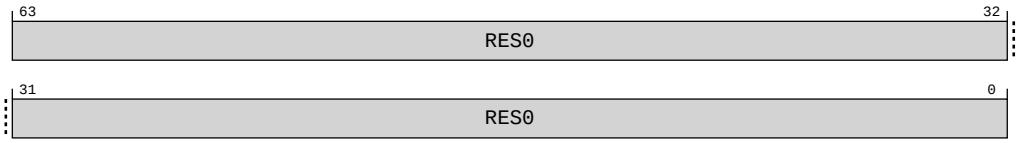


Table A-285: TRCIDR7 bit descriptions

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

Access

MRS <Xt>, TRCIDR7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1111	0b111

Accessibility

MRS <Xt>, TRCIDR7

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR7;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR7;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR7;
```

A.13.17 TRCDEVID, Device Configuration Register

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

AArch64 register TRCDEVID bits [31:0] are architecturally mapped to External System register B.7.27 TRCDEVID, Device Configuration Register on page 753 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Figure A-116: AArch64_trcdevid bit assignments

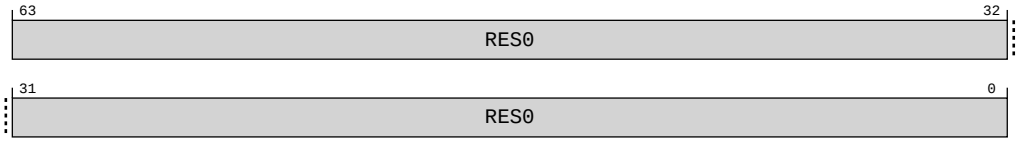


Table A-287: TRCDEVID bit descriptions

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

Access

MRS <Xt>, TRCDEVID

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b0010	0b111

Accessibility

MRS <Xt>, TRCDEVID

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCDEVID;
```

```
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCDEVID;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCDEVID;
```

A.13.18 TRCCLAIMSET, Claim Tag Set Register

In conjunction with AArch64-TRCCLAIMCLR, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information, see the CoreSight Architecture Specification.

Configurations

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.

AArch64 register TRCCLAIMSET bits [31:0] are architecturally mapped to External System register B.7.22 TRCCLAIMSET, Claim Tag Set Register on page 745 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

RAOW1S

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0000 0000 0000 0000 0000 1111

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-117: AArch64_trcclaimset bit assignments

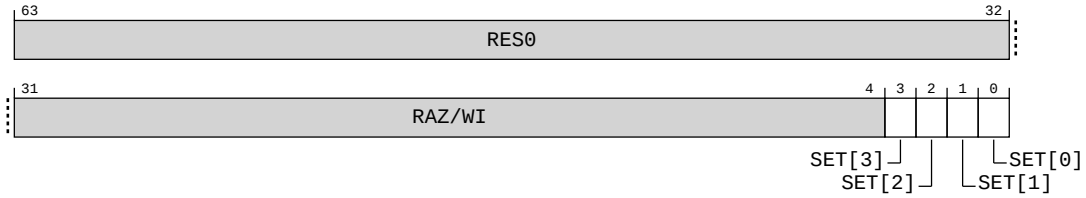


Table A-289: TRCCLAIMSET bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:4]	**RAZ/WI**	Reserved	**RAZ/WI**
[3]	SET[3]	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.	0b1
[2]	SET[2]	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.	0b1
[1]	SET[1]	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.	0b1

Bits	Name	Description	Reset
[0]	SET[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.	0b1

Access

MRS <Xt>, TRCCLAIMSET

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1000	0b110

MSR TRCCLAIMSET, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1000	0b110

Accessibility

MRS <Xt>, TRCCLAIMSET

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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.TRCCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCCLAIMSET;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCCLAIMSET;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
```

```
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCCLAIMSET;
```

MSR TRCCLAIMSET, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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.TRCCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCLAIMSET = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCLAIMSET = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCLAIMSET = X[t];
```

A.13.19 TRCCLAIMCLR, Claim Tag Clear Register

In conjunction with AArch64-TRCCLAIMSET, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information, see the CoreSight Architecture Specification.

Configurations

AArch64 register TRCCLAIMCLR bits [31:0] are architecturally mapped to External System register B.7.23 TRCCLAIMCLR, Claim Tag Clear Register on page 747 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

RW1C

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-118: AArch64_trcclaimclr bit assignments

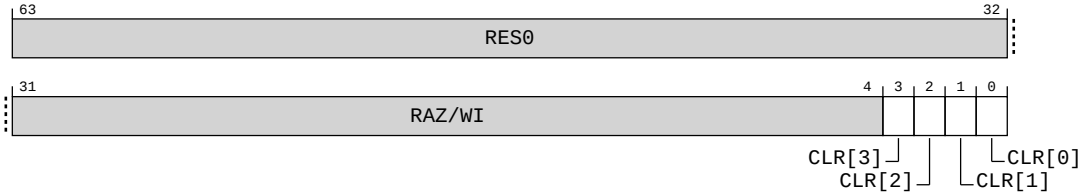


Table A-292: TRCCLAIMCLR bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:4]	**RAZ/WI**	Reserved	**RAZ/WI**
[3]	CLR[3]	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.	0b0

Bits	Name	Description	Reset
[2]	CLR[2]	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.	0b0
[1]	CLR[1]	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.	0b0
[0]	CLR[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.	0b0

Access

MRS <Xt>, TRCCLAIMCLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1001	0b110

MSR TRCCLAIMCLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1001	0b110

Accessibility

MRS <Xt>, TRCCLAIMCLR

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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.TRCCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCCLAIMCLR;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCCLAIMCLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCCLAIMCLR;
```

MSR TRCCLAIMCLR, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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.TRCCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCLAIMCLR = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCLAIMCLR = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCLAIMCLR = X[t];
```

A.13.20 TRCDEVARCH, Device Architecture Register

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

AArch64 register TRCDEVARCH bits [31:0] are architecturally mapped to External System register B.7.24 TRCDEVARCH, Device Architecture Register on page 749 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-119: AArch64_trcdevarch bit assignments

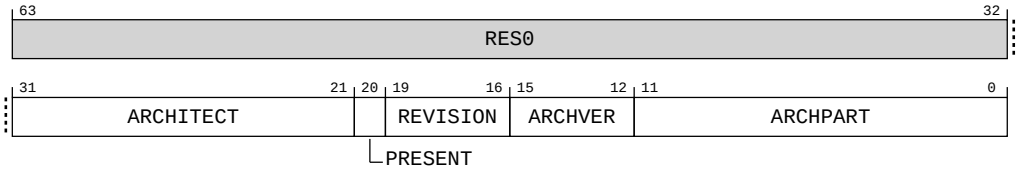


Table A-295: TRCDEVARCH bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code. **0b01000111011** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited. Other values are defined by the JEDEC JEP106 standard. This field reads as 0x23B.	11{x}
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. **0b1** Device Architecture information present.	x
[19:16]	REVISION	Revision. Defines the architecture revision of the component. **0b0001** ETEv1.1, FEAT_ETEv1p1.	xxxx
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **0b0101** ETEv1. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12]. This field reads as 0x5.	xxxx
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **0b101000010011** Arm PE trace architecture. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHPART is ARCHID[11:0]. This field reads as 0xA13.	12{x}

Access

MRS <Xt>, TRCDEVARCH

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1111	0b110

Accessibility

MRS <Xt>, TRCDEVARCH

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && 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 Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCDEVARCH;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCDEVARCH;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCDEVARCH;
```

A.14 AArch64 Trace Buffer Extension registers summary

The summary table provides an overview of all Trace Buffer Extension 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-297: Trace Buffer Extension registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRBLIMITR_EL1	3	0	C9	C11	0	–	64-bit	Trace Buffer Limit Address Register
TRBPTR_EL1	3	0	C9	C11	1	–	64-bit	Trace Buffer Write Pointer Register
TRBBASER_EL1	3	0	C9	C11	2	–	64-bit	Trace Buffer Base Address Register
TRBSR_EL1	3	0	C9	C11	3	–	64-bit	Trace Buffer Status/syndrome Register
TRBMAR_EL1	3	0	C9	C11	4	–	64-bit	Trace Buffer Memory Attribute Register
TRBTRG_EL1	3	0	C9	C11	6	–	64-bit	Trace Buffer Trigger Counter Register
TRBIDR_EL1	3	0	C9	C11	7	–	64-bit	Trace Buffer ID Register

A.14.1 TRBIDR_EL1, Trace Buffer ID Register

Describes constraints on using the Trace Buffer Unit to software, including whether the Trace Buffer Unit can be programmed at the current Exception level.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace Buffer Extension registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx1x 0110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-120: AArch64_trbidr_el1 bit assignments

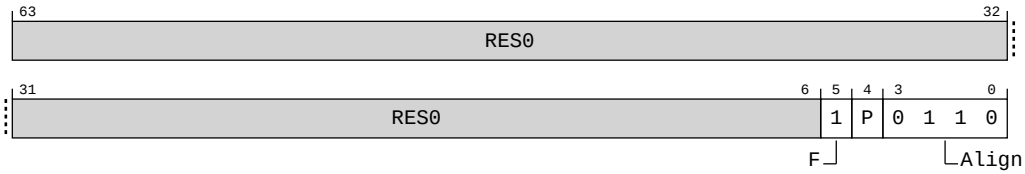


Table A-298: TRBIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:6]	RES0	Reserved	RES0
[5]	F	Flag updates. Describes how address translations performed by the Trace Buffer Unit manage the Access flag and dirty state. **0b1** Hardware management of the Access flag and dirty state for accesses made by the Trace Buffer Unit is controlled in the same way as explicit memory accesses in the trace buffer owning translation regime.	0b1

Bits	Name	Description	Reset
[4]	P	Programming not allowed. When read at EL3, this field reads as zero. Otherwise, indicates that the trace buffer is owned by a higher Exception level or another Security state. Defined values are: **0b0** Programming is allowed. **0b1** Programming not allowed. The value read from this field depends on the current Exception level and the Effective values of AArch64-MDCR_EL3.NSTB and AArch64-MDCR_EL2.E2TB: • If EL3 is implemented, and the owning Security state is Secure state, this field reads as one from: ◦ Non-secure EL1 and Non-secure EL2. ◦ If Secure EL2 is implemented and enabled, and AArch64-MDCR_EL2.E2TB is 0b00, Secure EL1. • If EL3 is implemented, and the owning Security state is Non-secure state, this field reads as one from: ◦ Secure EL1. ◦ If Secure EL2 is implemented, Secure EL2. ◦ If EL2 is implemented and AArch64-MDCR_EL2.E2TB is 0b00, Non-secure EL1. • If EL3 is not implemented, EL2 is implemented, and AArch64-MDCR_EL2.E2TB is 0b00, this field reads as one from EL1. • Otherwise, this field reads as zero.	x
[3:0]	Align	Defines the minimum alignment constraint for writes to AArch64-TRBPTR_EL1 and AArch64-TRBTRG_EL1. Defined values are: **0b0110** 64 bytes.	0b0110

Access

MRS <Xt>, TRBIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b111

Accessibility

MRS <Xt>, TRBIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRBIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return TRBIDR_EL1;
elsif PSTATE.EL == EL2 then
    return TRBIDR_EL1;
elsif PSTATE.EL == EL3 then
    return TRBIDR_EL1;
```

Appendix B External registers

This appendix contains the descriptions for the Cortex-A520 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 MPMM registers summary

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

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

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: MPMM registers summary

Offset	Name	Reset	Width	Description
0x000	CPUPPMCR	–	64-bit	Global PPM Configuration Register
0x010	CPUMPMMCR	–	64-bit	Global MPMM Configuration Register

B.1.1 CPUPPMCR, Global PPM Configuration Register

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

Configurations

External register CPUPPMCR bits [63:0] are architecturally mapped to AArch64 System register A.2.1 IMP_CPUPPMCR_EL3, Global PPM Configuration Register on page 236 bits [63:0].

Attributes

Width

64

Component

MPMM

Register offset

0x000

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-1: ext_cpuppmcr bit assignments

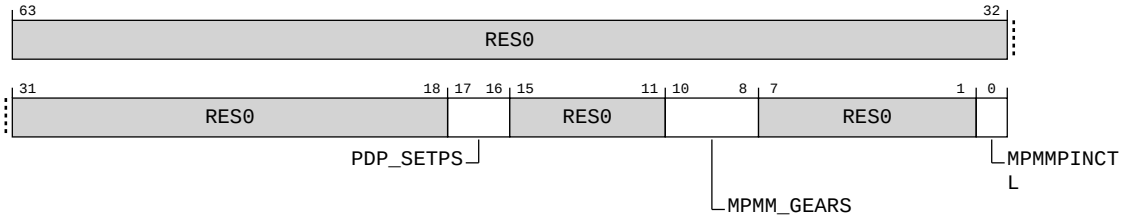


Table B-2: CPUPPMCR bit descriptions

Bits	Name	Description	Reset
[63:18]	**RES0**	Reserved	**RES0**
[17:16]	PDP_SETPS	Number of PDP Setpoints implemented **0b00** PDP is not implemented or enabled. Access to this field is: RO	xx
[15:11]	**RES0**	Reserved	**RES0**
[10:8]	MPMM_GEARS	Number of MPMM Gears implemented **0b011** 3 MPMM are enabled. Access to this field is: RO	xxx
[7:1]	**RES0**	Reserved	**RES0**
[0]	MPMMPINCTL	MPMM Pin Control Enabled **0b0** MPMM control through SPR and utility bus. **0b1** MPMM control through pin only.	0b0

Accessibility

Component	Offset	Instance	Range
MPMM	0x000	CPUPPMCR	None

This interface is accessible as follows:

When IsCorePowered() && IsAccessSecure()
RW

When IsCorePowered() && !IsAccessSecure()
RAZ/WI

Otherwise
ERROR

B.1.2 CPUMPMMCR, Global MPMM Configuration Register

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

Configurations

External register CPUMPMMCR bits [63:0] are architecturally mapped to AArch64 System register A.5.25 IMP_CPUMPMMCR_EL3, Global MPMM Configuration Register on page 325 bits [63:0].

Attributes

Width
64

Component
MPMM

Register offset
0x010

Access type
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x000


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-2: ext_cpumpmmcr bit assignments

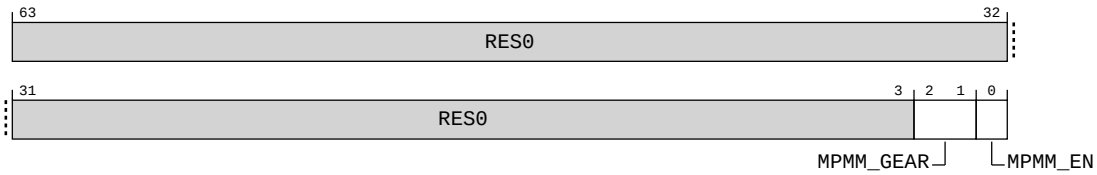


Table B-4: CPUMPMMCR bit descriptions

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.	0b00
[0]	MPMM_EN	MPMM Master Enable **0b0** MPMM is disabled. **0b1** MPMM is enabled.	0b0

Accessibility

Component	Offset	Instance	Range
MPMM	0x010	CPUMPMMCR	None

This interface is accessible as follows:

When IsCorePowered() && IsAccessSecure()

RW

When IsCorePowered() && !IsAccessSecure()

RAZ/WI

Otherwise

ERROR

B.2 External Complex RAS registers summary

The summary table provides an overview of all memory-mapped Complex 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 B-6: Complex RAS registers summary

Offset	Name	Reset	Width	Description
0x0	ERR0FR	—	64-bit	Error Record Feature Register
0x8	ERR0CTLR	—	64-bit	Error Record Control Register
0x10	ERR0STATUS	—	64-bit	Error Record Primary Status Register
0x20	ERR0MISC0	—	64-bit	Error Record Miscellaneous Register 0
0x28	ERR0MISC1	—	64-bit	Error Record Miscellaneous Register 1
0x30	ERR0MISC2	—	64-bit	Error Record Miscellaneous Register 2
0x38	ERR0MISC3	—	64-bit	Error Record Miscellaneous Register 3
0x800	ERR0PFGF	—	64-bit	Pseudo-fault Generation Feature Register
0x808	ERR0PFGCTL	—	64-bit	Pseudo-fault Generation Control Register
0x810	ERR0PFGCDN	—	64-bit	Pseudo-fault Generation Countdown Register
0xE00	ERRGSR	—	64-bit	Error Group Status Register
0xE10	ERRIIDR	—	32-bit	Implementation Identification Register
0xFA8	ERRDEVAFF	—	64-bit	Device Affinity Register
0xFBC	ERRDEVARCH	—	32-bit	Device Architecture Register
0xFC8	ERRDEVID	—	32-bit	Device Configuration Register
0xFD0	ERRPIDR4	—	32-bit	Peripheral Identification Register 4
0xFE0	ERRPIDR0	—	32-bit	Peripheral Identification Register 0
0xFE4	ERRPIDR1	—	32-bit	Peripheral Identification Register 1
0xFE8	ERRPIDR2	—	32-bit	Peripheral Identification Register 2
0xFEC	ERRPIDR3	—	32-bit	Peripheral Identification Register 3
0xFF0	ERRCIDR0	—	32-bit	Component Identification Register 0
0xFF4	ERRCIDR1	—	32-bit	Component Identification Register 1
0xFF8	ERRCIDR2	—	32-bit	Component Identification Register 2
0xFFC	ERRCIDR3	—	32-bit	Component Identification Register 3

B.2.1 ERR0FR, Error Record Feature Register

Defines whether <n> is the first record owned by a node:

- If <n> is the first error record owned by a node, then ERR<n>FR.ED is not 0b00.
- If <n> is not the first error record owned by a node, then ERR<n>FR.ED is 0b00.

If <n> is the first record owned by the node, defines which of the common architecturally-defined features are implemented by the node and, of the implemented features, which are software programmable.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Complex RAS

Register offset

0x0

Access type

RO

Reset value

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

Figure B-3: complex_ras.err0fr bit assignments

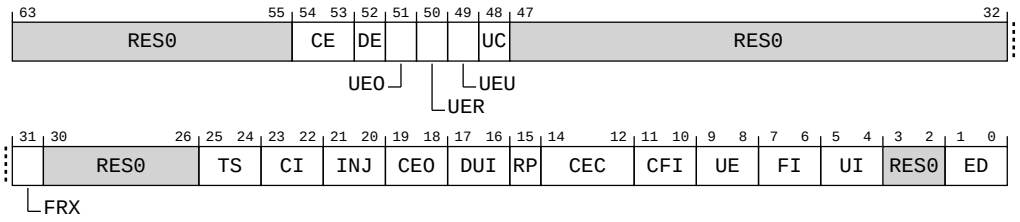


Table B-7: ERR0FR bit descriptions

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. **0b00** Does not record Corrected Errors. This value is reported if CPU_CACHE_PROTECTION is disabled **0b10** Records non-specific Corrected Errors. This value is reported if CPU_CACHE_PROTECTION is enabled	xx
[52]	DE	Deferred Error recording. Describes whether the node supports recording Deferred errors. **0b0** Does not record Deferred Errors. This value is reported if CPU_CACHE_PROTECTION is disabled **0b1** Records Deferred Errors. This value is reported if CPU_CACHE_PROTECTION is enabled	x
[51]	UEO	Latent or Restartable Error recording. Describes whether the node supports recording Latent or Restartable errors. **0b1** Records Latent or Restartable errors.	x
[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.	x
[49]	UEU	Unrecoverable Error recording. Describes whether the node supports recording Unrecoverable errors. **0b0** Does not record Unrecoverable errors.	x
[48]	UC	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. **0b1** Records Uncontainable errors.	x
[47:32]	**RES0**	Reserved	**RES0**
[31]	FRX	Feature Register extension. Defines whether ERR<n>FR[63:48] are architecturally defined. **0b1** ERR<n>FR[63:48] are defined by the architecture.	x
[30:26]	**RES0**	Reserved	**RES0**
[25:24]	TS	Timestamp Extension. Indicates whether, for each error record <m> owned by this node, ERR<m>MISC3 is used as the timestamp register, and, if it is, the timebase used by the timestamp. **0b00** Does not support a timestamp register.	xx
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented by the node. **0b00** Does not support the critical error interrupt. ext-ERR<n>CTLR.CI is **RES0**.	xx

Bits	Name	Description	Reset
[21:20]	INJ	Fault Injection Extension. Indicates whether the Common Fault Injection Model Extension is implemented by the node. **0b00** Does not support the Common Fault Injection Model Extension. This value is reported if CPU_CACHE_PROTECTION is disabled **0b01** Supports the Common Fault Injection Model Extension. See ERR<n>PFGF for more information. This value is reported if CPU_CACHE_PROTECTION is enabled	xx
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior of the node when a second or subsequent Corrected error is recorded and a first Corrected error has previously been recorded by an error record <m> owned by the node. **0b00** Keeps the previous error syndrome.	xx
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the enabling and disabling of error recovery interrupts on deferred errors is supported by the node. **0b10** Enabling and disabling of error recovery interrupts on deferred errors is supported and controllable using ext-ERR<n>CTLR.DUI.	xx
[15]	RP	Repeat counter. Indicates whether the node implements a second Corrected error counter in ERR<m>MISC0 for each error record <m> owned by the node that can record countable errors. **0b1** Implements a first (repeat) counter and a second (other) counter in ERR<m>MISC0 for each error record <m> owned by the node that can record countable errors. The repeat counter is the same size as the primary error counter.	x
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter mechanisms in ERR<m>MISC0 for each error record <m> owned by the node that can record countable errors. **0b010** Implements an 8-bit Corrected error counter in ERR<m>MISC0[39:32] for each error record <m> owned by the node that can record countable errors.	xxx
[11:10]	CFI	Fault handling interrupt for corrected errors control. Indicates whether the enabling and disabling of fault handling interrupts on corrected errors is supported by the node. **0b10** Enabling and disabling of fault handling interrupts on corrected errors is supported and controllable using ext-ERR<n>CTLR.CFI.	xx
[9:8]	UE	In-band error reponse (External Abort). Indicates whether the in-band error response and associated controls are implemented by the node. **0b01** In-band error response is supported and always enabled. ext-ERR<n>CTLR.UE is **RES0**.	xx
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented by the node. **0b10** Fault handling interrupt is supported and controllable using ext-ERR<n>CTLR.FI.	xx
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented by the node. **0b10** Error handling interrupt is supported and controllable using ext-ERR<n>CTLR.UI.	xx
[3:2]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[1:0]	ED	Error reporting and logging. Indicates error record <n> is the first record owned the node, and whether the node implements the controls for enabling and disabling error reporting and logging. **0b10** Error reporting and logging is controllable using ext-ERR<n>CTLR.ED.	xx

Accessibility

Component	Offset	Instance	Range
Complex RAS	0x0	ERR0FR	None

This interface is accessible as follows:

RO

B.2.2 ERR0CTLR, Error Record Control Register

The error control register contains enable bits for the node that writes to this record:

- Enabling error detection and correction.
- Enabling the critical error, error recovery, and fault handling interrupts.
- Enabling in-band error response for uncorrected errors.

For each bit, if the node does not support the feature, then the bit is **RES0**. The definition of each record is IMPLEMENTATION DEFINED.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Complex RAS

Register offset

0x8

Access type

RW

Reset value

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

Figure B-4: complex_ras.err0ctlr bit assignments

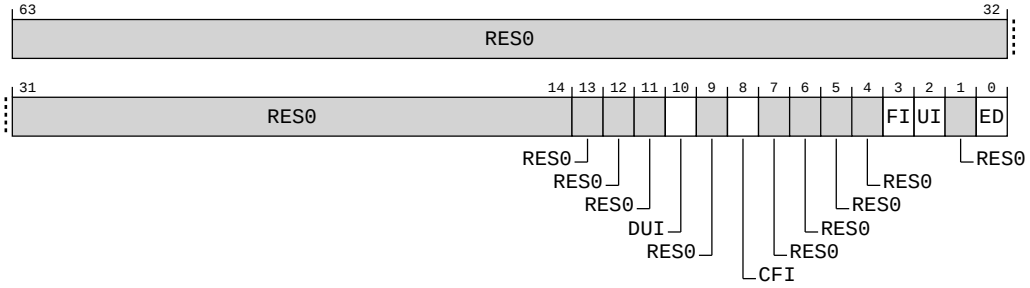


Table B-9: ERR0CTLR bit descriptions

Bits	Name	Description	Reset
[63:11]	**RES0**	Reserved	**RES0**
[10]	DUI	Error recovery interrupt for Deferred errors enable. When ext-ERR<n>FR.DUI == `0b10`, this control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all errors recorded as Deferred error. **`0b0`** Error recovery interrupt not generated for Deferred errors. **`0b1`** Error recovery interrupt generated for Deferred errors. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	x
[9]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[8]	CFI	Fault handling interrupt for Corrected errors enable. When ext-ERR<n>FR.CFI == `0b10`, this control applies to errors arising from both reads and writes. When enabled: • If the node implements Corrected error counters, then the fault handling interrupt is generated when a counter overflows and the overflow bit for the counter is set to 1. For more information, see ext-ERR<n>MISC0. • Otherwise, the fault handling interrupt is also generated for all errors recorded as Corrected error. **0b0** Fault handling interrupt not generated for Corrected errors. **0b1** Fault handling interrupt generated for Corrected errors. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	x
[7:4]	**RES0**	Reserved	**RES0**
[3]	FI	Fault handling interrupt enable. When ext-ERR<n>FR.FI == `0b10`, this control applies to errors arising from both reads and writes. When enabled: • The fault handling interrupt is generated for all errors recorded as either Deferred error or Uncorrected error. • If the fault handling interrupt for Corrected errors control is not implemented: ◦ If the node implements Corrected error counters, then the fault handling interrupt is also generated when a counter overflows and the overflow bit for the counter is set to 1. ◦ Otherwise, the fault handling interrupt is also generated for all errors recorded as Corrected error. **0b0** Fault handling interrupt disabled. **0b1** Fault handling interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	x
[2]	UI	Uncorrected error recovery interrupt enable. When ext-ERR<n>FR.UI == `0b10`, this control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all errors recorded as Uncorrected error. **0b0** Error recovery interrupt disabled. **0b1** Error recovery interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	x
[1]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[0]	ED	Error reporting and logging enable. When disabled, the node behaves as if error detection and correction are disabled, and no errors are recorded or signaled by the node. Arm recommends that, when disabled, correct error detection and correction codes are written for writes, unless disabled by an **IMPLEMENTATION DEFINED** control for error injection. **0b0** Error reporting disabled. **0b1** Error reporting enabled. It is IMPLEMENTATION DEFINED whether the node fully disables error detection and correction when reporting is disabled. That is, even with error reporting disabled, the node might continue to silently correct errors. Uncorrected errors might result in corrupt data being silently propagated by the node. **Note:** If this node requires initialization after Cold reset to prevent signaling false errors, then Arm recommends this field is set to 0 on Cold reset, meaning errors are not reported from Cold reset. This allows boot software to initialize a node without signaling errors. Software can enable error reporting after the node is initialized. Otherwise, the Cold reset value is IMPLEMENTATION DEFINED. If the Cold reset value is 1, the reset values of other controls in this register are also IMPLEMENTATION DEFINED and should not be **UNKNOWN**.	x

Accessibility

Component	Offset	Instance	Range
Complex RAS	0x8	ERR0CTLR	None

This interface is accessible as follows:

RW

B.2.3 ERR0STATUS, Error Record Primary Status Register

Contains status information for error record <n>, 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:

- ERR<n>STATUS.{AV, V, MV} are valid bits that define whether error record <n> registers are valid.
- ERR<n>STATUS.{UE, OF, CE, DE, UET} encode the types of error or errors recorded.
- ERR<n>STATUS.{CI, ER, PN, IERR, SERR} are syndrome fields.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Complex RAS

Register offset

0x10

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x0xx x0xx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-5: complex_ras.err0status bit assignments

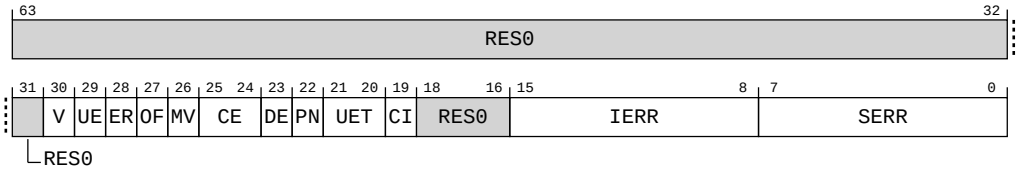


Table B-11: ERR0STATUS bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[30]	V	Status Register Valid. **0b0** ERR<n>STATUS not valid. **0b1** ERR<n>STATUS valid. At least one error has been recorded. Access to this field is: W1C	0b0
[29]	UE	Uncorrected Error. **0b0** No errors have been detected, or all detected errors have been either corrected or deferred. **0b1** At least one detected error was not corrected and not deferred. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x

Bits	Name	Description	Reset
[28]	ER	Error Reported. **0b0** No in-band error response (External Abort) signaled to the Requester making the access or other transaction. **0b1** An in-band error response was signaled by the component to the Requester making the access or other transaction. This can be because any of the following are true: • The applicable one of the ERR<q>CTLR.{WUE, RUE, UE} fields is implemented and was 1 when an error was detected and not corrected. • The applicable one of the ERR<q>CTLR.{WUE, RUE, UE} fields is not implemented and the component always reports errors. It is **IMPLEMENTATION DEFINED** whether an uncorrected error that is deferred and recorded as a Deferred error, but is not deferred to the Requester, will signal an in-band error response to the Requester, causing this field to be set to 1. If no in-band error response to the Requester, this field is set to 0. **Note:** An in-band error response signaled by the component might be masked and not generate any exception. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.UE == '0' && ext-ERR0STATUS.DE == '0' && this field can be set to 0b1 by a Deferred error** Access to this field is: **UNKNOWN**/WI **When ext-ERR0STATUS.UE == '0' && this field is never set to 0b1 by a Deferred error** Access to this field is: **UNKNOWN**/WI **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x

Bits	Name	Description	Reset
[27]	OF	Overflow. Indicates that multiple errors have been detected. This field is set to 1 when one of the following occurs: • A Corrected error counter is implemented, an error is counted, and the counter overflows. • ERR<n>STATUS.V was previously 1, a Corrected error counter is not implemented, and a Corrected error is recorded. • ERR<n>STATUS.V was previously 1, and a type of error other than a Corrected error is recorded. Otherwise, this field is unchanged when an error is recorded. If a Corrected error counter is implemented: • A direct write that modifies the counter overflow flag indirectly might set this field to an **UNKNOWN** value. • A direct write to this field that clears this field to zero might indirectly set the counter overflow flag to an **UNKNOWN** value. **0b0** Since this field was last cleared to zero, no error syndrome has been discarded and, if a Corrected error counter is implemented, it has not overflowed. **0b1** Since this field was last cleared to zero, at least one error syndrome has been discarded or, if a Corrected error counter is implemented, it might have overflowed. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x
[26]	MV	Miscellaneous Registers Valid. **0b0** ERR<n>MISC<m> not valid. **0b1** The **IMPLEMENTATION DEFINED** contents of the ERR<n>MISC<m> registers contains additional information for an error recorded by this record. **Note:** If the ERR<n>MISC<m> registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded. Access to this field is: W1C	0b0

Bits	Name	Description	Reset
[25:24]	CE	Corrected Error. **0b00** No errors were corrected. **0b10** At least one error was corrected. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	xx
[23]	DE	Deferred Error. **0b0** No errors were deferred. **0b1** At least one error was not corrected and deferred. Support for deferring errors is **IMPLEMENTATION DEFINED**. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x
[22]	PN	Poison. **0b0** Uncorrected error or Deferred error recorded because a corrupt value was detected, for example, by an error detection code (EDC), or Corrected error recorded. **0b1** Uncorrected error or Deferred error recorded because a poison value was detected. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0' || (ext-ERR0STATUS.DE == '0' && ext-ERR0STATUS.UE == '0')** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x

Bits	Name	Description	Reset
[21:20]	UET	Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. **0b00** Uncorrected error, Uncontainable error (UC). **0b01** Uncorrected error, Unrecoverable error (UEU). **0b10** Uncorrected error, Latent or Restartable error (UEO). **0b11** Uncorrected error, Signaled or Recoverable error (UER). **Note:** Software might use the information in the error record registers to determine what recovery is necessary. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0' || ext-ERR0STATUS.UE == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	xx
[19]	CI	Critical Error. Indicates whether a critical error condition has been recorded. **0b0** No critical error condition. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x
[18:16]	**RES0**	Reserved	**RES0**
[15:8]	IERR	**IMPLEMENTATION DEFINED** error code. Used with any primary error code ERR<n>STATUS.SERR value. Further **IMPLEMENTATION DEFINED** information can be placed in the ERR<n>MISC<m> registers. The implemented set of valid values that this field can take is IMPLEMENTATION DEFINED. If any value not in this set is written to this register, then the value read back from this field is **UNKNOWN**. **Note:** This means that one or more bits of this field might be implemented as fixed read-as-zero or read-as-one values. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RW	8{x}

Bits	Name	Description	Reset
[7:0]	SERR	Architecturally-defined primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry. **0b00000110** Data value from associative memory. For example, ECC error on cache data. **0b00000111** Address/control value from associative memory. For example, ECC error on cache tag. **0b00001000** Data value from a TLB. For example, ECC error on TLB data. **0b00001100** Data value from (non-associative) external memory. For example, ECC error in SDRAM. **0b00010010** Error response from Completer of access. For example, error response from cache write-back. **0b00010101** Deferred error from Completer not supported at Requester. For example, poisoned data received from the Completer of an access by a Requester that cannot defer the error further. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RW	8{x}

Access

ERR<n>STATUS.{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. ERR<n>STATUS.{IERR, SERR} are read/write (RW) fields, although the set of implemented valid values is **IMPLEMENTATION DEFINED**. See also ext-ERR<n>PFGF.SYN.

After reading ERR<n>STATUS, 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 ERR<n>STATUS.{UE, DE, CE} are ignored if ERR<n>STATUS.OF is 1 and is not being cleared to 0.
- Writes to ERR<n>STATUS.V are ignored if any of ERR<n>STATUS.{UE, DE, CE} are nonzero and are not being cleared to zero.
- Writes to ERR<n>STATUS.{AV, MV} and the ERR<n>STATUS.{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 ERR<n>STATUS.UE, ERR<n>STATUS.DE, and ERR<n>STATUS.CE.

When RAS System Architecture v1.1 is implemented, a write to the register is ignored if all of:

- Any of ERR<n>STATUS.{V, UE, OF, CE, DE} are nonzero before the write.

- The write does not clear the nonzero ERR<n>STATUS.{V, UE, OF, CE, DE} fields to zero by writing ones to the applicable field or fields.

Some of the fields in ERR<n>STATUS are also defined as **UNKNOWN** where certain combinations of ERR<n>STATUS.{V, DE, UE} are zero. The rules for writes to ERR<n>STATUS 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 ERR<n>STATUS when ERR<n>STATUS.V is 1 results in either ERR<n>STATUS.V field being cleared to zero, or ERR<n>STATUS.V not changing. Since all fields in ERR<n>STATUS, other than ERR<n>STATUS.{AV, V, MV}, usually read as **UNKNOWN** values when ERR<n>STATUS.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:

- Read ERR<n>STATUS and determine which fields need to be cleared to zero.
- 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.

Otherwise, these fields might not have the correct value when a new fault is recorded.

An exception is when the node supports writing to these fields as part of fault injection. See also ext-ERR<n>PFGF.SYN.

Accessibility

ERR<n>STATUS.{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. ERR<n>STATUS.{IERR, SERR} are read/write (RW) fields, although the set of implemented valid values is IMPLEMENTATION DEFINED. See also ext-ERR<n>PFGF.SYN.

After reading ERR<n>STATUS, 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 ERR<n>STATUS.{UE, DE, CE} are ignored if ERR<n>STATUS.OF is 1 and is not being cleared to 0.
- Writes to ERR<n>STATUS.V are ignored if any of ERR<n>STATUS.{UE, DE, CE} are nonzero and are not being cleared to zero.
- Writes to ERR<n>STATUS.{AV, MV} and the ERR<n>STATUS.{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 ERR<n>STATUS.UE, ERR<n>STATUS.DE, and ERR<n>STATUS.CE.

When RAS System Architecture v1.1 is implemented, a write to the register is ignored if all of:

- Any of ERR<n>STATUS.{V, UE, OF, CE, DE} are nonzero before the write.
- The write does not clear the nonzero ERR<n>STATUS.{V, UE, OF, CE, DE} fields to zero by writing ones to the applicable field or fields.

Some of the fields in ERR<n>STATUS are also defined as UNKNOWN where certain combinations of ERR<n>STATUS.{V, DE, UE} are zero. The rules for writes to ERR<n>STATUS 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 ERR<n>STATUS when ERR<n>STATUS.V is 1 results in either ERR<n>STATUS.V field being cleared to zero, or ERR<n>STATUS.V not changing. Since all fields in ERR<n>STATUS, other than ERR<n>STATUS.{AV, V, MV}, usually read as UNKNOWN values when ERR<n>STATUS.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:

- Read ERR<n>STATUS and determine which fields need to be cleared to zero.
- 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.

Otherwise, these fields might not have the correct value when a new fault is recorded.

An exception is when the node supports writing to these fields as part of fault injection. See also ext-ERR<n>PFGF.SYN.

Component	Offset	Instance	Range
Complex RAS	0x10	ERR0STATUS	None

This interface is accessible as follows:

When ext-ERR<n>STATUS.V != '0' && ERR<n>STATUS.V is not being cleared to 0b0 in the same write

RO

When ext-ERR<n>STATUS.UE != '0' && ERR<n>STATUS.UE is not being cleared to 0b0 in the same write

RO

When ext-ERR<n>STATUS.OF != '0' && ERR<n>STATUS.OF is not being cleared to 0b0 in the same write

RO

When ext-ERR<n>STATUS.CE != '00' && ERR<n>STATUS.CE is not being cleared to 0b00 in the same write

RO

When ext-ERR<n>STATUS.DE != '0' && ERR<n>STATUS.DE is not being cleared to 0b0 in the same write

RO

Otherwise

RW

B.2.4 ERR0MISC0, Error Record Miscellaneous Register 0

Contains information recording the cache line or TLB entry of a detected RAM error. Also contains the architecturally-defined Corrected error counters.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Complex RAS

Register offset

0x20

Access type

See bit descriptions

Reset value

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

Figure B-6: complex_ras.err0misc0 bit assignments

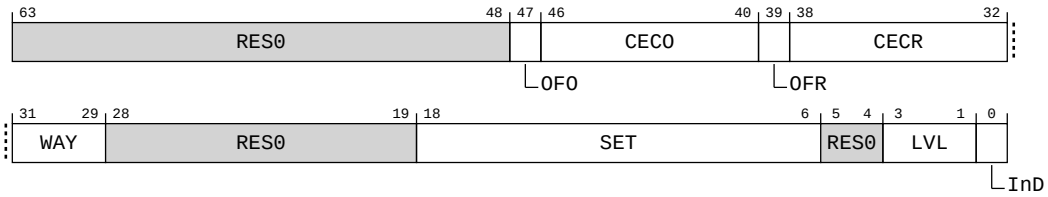


Table B-13: ERR0MISC0 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERR<n>MISC0.CECO is incremented and wraps through zero. **0b0** Other counter has not overflowed. **0b1** Other counter has overflowed. A direct write that modifies this field might indirectly set ext-ERR<n>STATUS.OF to an **UNKNOWN** value and a direct write to ext-ERR<n>STATUS.OF that clears it to zero might indirectly set this field to an **UNKNOWN** value.	x
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERR<n>MISC0.CECR.	7{x}
[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERR<n>MISC0.CECR is incremented and wraps through zero. **0b0** Repeat counter has not overflowed. **0b1** Repeat counter has overflowed. A direct write that modifies this field might indirectly set ext-ERR<n>STATUS.OF to an **UNKNOWN** value and a direct write to ext-ERR<n>STATUS.OF that clears it to zero might indirectly set this field to an **UNKNOWN** value.	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. **Note:** For example, the other syndrome might include the set and way information for an error detected in a cache. This might be recorded in the **IMPLEMENTATION DEFINED** ERR<n>MISC<m> fields on a first Corrected error. ERR<n>MISC0.CECR is then incremented for each subsequent Corrected Error in the same set and way.	7{x}
[31:29]	WAY	The way that contained the error If the encoding of the way for the reported RAM requires fewer bits than the width of this field, the most significant bits of this field record the way, and the least significant bits are **RES0**. **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	xxx
[28:19]	**RES0**	Reserved	**RES0**
[18:6]	SET	The set that contained the error **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	13{x}
[5:4]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[3:1]	LVL	Cache level **0b000** L1. **0b001** L2. **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	xxx
[0]	InD	Instruction or Data cache **0b0** Data or unified cache. **0b1** Instruction cache. **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	x

Access

Reads from ERR<n>MISC0 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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.

Accessibility

Reads from ERR<n>MISC0 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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
Complex RAS	0x20	ERR0MISC0	None

This interface is accessible as follows:

RW

B.2.5 ERR0MISC1, Error Record Miscellaneous Register 1

Contains information recording the exact location of a detected RAM error. Refer to the Cortex-A520 Core Configuration and Integration Manual to interpret the fields in this register.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Complex RAS

Register offset

0x28

Access type

See bit descriptions

Reset value

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

Figure B-7: complex_ras.err0misc1 bit assignments

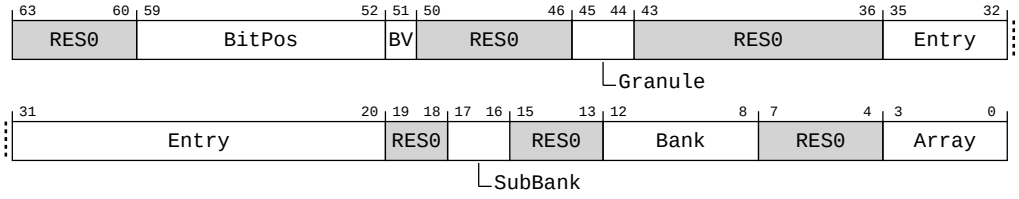


Table B-15: ERR0MISC1 bit descriptions

Bits	Name	Description	Reset
[63:60]	**RES0**	Reserved	**RES0**
[59:52]	BitPos	The bit position that contained the error **When ext-ERR0MISC1.BV == '0'** Access to this field is: **RES0** **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	8{x}
[51]	BV	Indicates that the BitPos field contains valid data **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	x
[50:46]	**RES0**	Reserved	**RES0**
[45:44]	Granule	The protection granule within the ram entry containing the error **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	xx
[43:36]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[35:20]	Entry	The RAM row containing the error **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	16{x}
[19:18]	**RES0**	Reserved	**RES0**
[17:16]	SubBank	The sub-bank of the RAM bank containing the error **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	xx
[15:13]	**RES0**	Reserved	**RES0**
[12:8]	Bank	The RAM bank within the array containing the error **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	5{x}
[7:4]	**RES0**	Reserved	**RES0**
[3:0]	Array	The specific RAM array containing the error **0b1000** L2 cache data RAMs. **0b1001** L2 cache tag RAMs. **0b1010** L2 cache L2DB RAMs. **0b1011** L2 cache duplicate L1 D-cache tag RAMs. **0b1100** L2 TLB RAMs. **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	xxxx

Access

Reads from ERR<n>MISC1 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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.

Accessibility

Reads from ERR<n>MISC1 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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
Complex RAS	0x28	ERR0MISC1	None

This interface is accessible as follows:

RW

B.2.6 ERR0MISC2, Error Record Miscellaneous Register 2

This register is not used and is reserved.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Complex RAS

Register offset

0x30

Access type

Read

R

Write

W

Reset value

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

Figure B-8: complex_ras.err0misc2 bit assignments

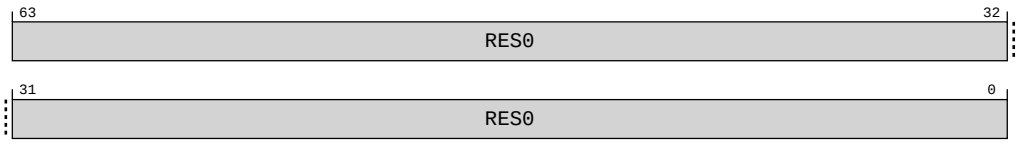


Table B-17: ERR0MISC2 bit descriptions

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

Access

Reads from ERR<n>MISC2 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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.

Accessibility

Reads from ERR<n>MISC2 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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
Complex RAS	0x30	ERR0MISC2	None

This interface is accessible as follows:

RW

B.2.7 ERR0MISC3, Error Record Miscellaneous Register 3

This register is not used and is reserved.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Complex RAS

Register offset

0x38

Access type

Read

R

Write

W

Reset value

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

Figure B-9: complex_ras.err0misc3 bit assignments

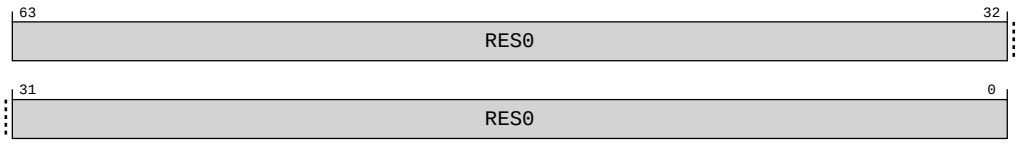


Table B-19: ERR0MISC3 bit descriptions

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

Access

Reads from ERR<n>MISC3 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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.

Accessibility

Reads from ERR<n>MISC3 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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
Complex RAS	0x38	ERR0MISC3	None

This interface is accessible as follows:

RW

B.2.8 ERR0PFGF, 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

Complex RAS

Register offset

0x800

Access type

RO

Reset value

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

Figure B-10: complex_ras.err0pfgf bit assignments

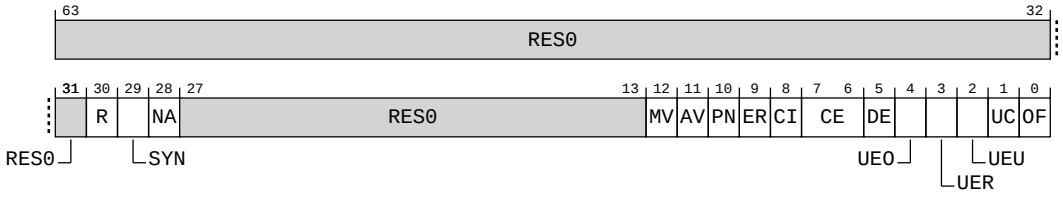


Table B-21: ERR0PFGF bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30]	R	Restartable. Support for Error Generation Counter restart mode. **0b0** Error Generation Counter restart mode is not supported. This value is reported if CPU_CACHE_PROTECTION is disabled **0b1** Error Generation Counter restart mode is implemented. This value is reported if CPU_CACHE_PROTECTION is enabled	x
[29]	SYN	Syndrome. Fault syndrome injection. **0b0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.{IERR, SERR} to IMPLEMENTATION DEFINED values. ext-ERR<n>STATUS.{IERR, SERR} are **UNKNOWN** when ext-ERR<n>STATUS.V is 0.	x

Bits	Name	Description	Reset
[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.	x
[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. It is **IMPLEMENTATION DEFINED** which ERR<n>MISC<m> syndrome fields, if any, are updated by the node when an injected error is recorded. Some syndrome fields might always be updated by the node when an error, including an injected error, is recorded. For example, a corrected error counter might always be updated when any countable error, including a injected countable error, is recorded. **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 ext-ERR<n>STATUS.MV is set to 1. • Otherwise, ext-ERR<n>STATUS.MV is unchanged. If the node always sets ext-ERR<n>STATUS.MV to 1 when recording an injected error then ext-ERR<n>PFGCTL.MV might be RAO/WI. Otherwise ext-ERR<n>PFGCTL.MV is **RES0**.	x
[11]	AV	Address syndrome. Defines whether software can control the address recorded in ext-ERR<n>ADDR when an injected error is recorded. **0b0** When an injected error is recorded, the node might record an address in ext-ERR<n>ADDR. If an address is recorded in ext-ERR<n>ADDR, then ext-ERR<n>STATUS.AV is set to 1. Otherwise, ext-ERR<n>ADDR and ext-ERR<n>STATUS.AV are unchanged. If the node always records an address and sets ext-ERR<n>STATUS.AV to 1 when recording an injected error then ext-ERR<n>PFGCTL.AV might be RAO/WI. Otherwise ext-ERR<n>PFGCTL.AV is **RES0**.	x
[10]	PN	Poison flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.PN status flag. **0b0** When an injected error is recorded, it is IMPLEMENTATION DEFINED whether the node sets ext-ERR<n>STATUS.PN to 1. ext-ERR<n>PFGCTL.PN is **RES0**.	x
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.ER status flag. **0b0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.ER according to the architecture-defined rules for setting the ER field. ext-ERR<n>PFGCTL.ER is **RES0**.	x
[8]	CI	Critical Error flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.CI status flag. **0b0** When an injected error is recorded, it is IMPLEMENTATION DEFINED whether the node sets ext-ERR<n>STATUS.CI to 1. ext-ERR<n>PFGCTL.CI is **RES0**.	x

Bits	Name	Description	Reset
[7:6]	CE	Corrected Error generation. Describes the types of Corrected error that the fault generation feature of the node can generate. **0b00** The fault generation feature of the node does not generate Corrected errors. This value is reported if CPU_CACHE_PROTECTION is disabled **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 ERR<n>STATUS.CE == 0b10. This value is reported if CPU_CACHE_PROTECTION is enabled	xx
[5]	DE	Deferred Error generation. Describes whether the fault generation feature of the node can generate Deferred errors. **0b0** The fault generation feature of the node does not generate Deferred errors. This value is reported if CPU_CACHE_PROTECTION is disabled **0b1** The fault generation feature of the node allows generation of Deferred errors. This value is reported if CPU_CACHE_PROTECTION is enabled	x
[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. ext-ERR<n>PFGCTL.UEO is **RES0**.	x
[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. ext-ERR<n>PFGCTL.UER is **RES0**.	x
[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. ext-ERR<n>PFGCTL.UEU is **RES0**.	x
[1]	UC	Uncontainable Error generation. Describes whether the fault generation feature of the node can generate Uncontainable errors. **0b0** The fault generation feature of the node does not generate Uncontainable errors. This value is reported if CPU_CACHE_PROTECTION is disabled **0b1** The fault generation feature of the node allows generation of Uncontainable errors. This value is reported if CPU_CACHE_PROTECTION is enabled	x
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.OF status flag. **0b0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.OF according to the architecture-defined rules for setting the OF field. ext-ERR<n>PFGCTL.OF is **RES0**.	x

Accessibility

Component	Offset	Instance	Range
Complex RAS	0x800	ERR0PFGF	None

This interface is accessible as follows:

RO

B.2.9 ERR0PFGCTL, Pseudo-fault Generation Control Register

Enables controlled fault generation.

Configurations

ext-ERR<n>FR and ext-ERR<n>PFGF describe the features implemented by the node.

Attributes

Width

64

Component

Complex RAS

Register offset

0x808

Access type

RW

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-11: complex_ras.err0pfgctl bit assignments

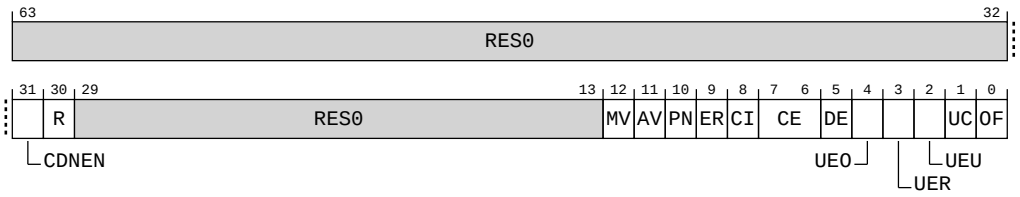


Table B-23: ERR0PFGCTL bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	CDNEN	Countdown Enable. Controls transfers of the value held in ext-ERR<n>PFGCDN to the Error Generation Counter and enables this counter. **0b0** The Error Generation Counter is disabled. This value (**RES0**) is reported if CPU_CACHE_PROTECTION is disabled **0b1** The Error Generation Counter is enabled. On a write of 1 to this field, the Error Generation Counter is set to ext-ERR<n>PFGCDN.CDN	0b0
[30]	R	Restart. Controls whether the Error Generation Counter restarts or stops counting on reaching zero. **0b0** On reaching zero, the Error Generation Counter will stop counting. This value (**RES0**) is reported if CPU_CACHE_PROTECTION is disabled **0b1** On reaching zero, the Error Generation Counter is set to ext-ERR<n>PFGCDN.CDN	x
[29:13]	**RES0**	Reserved	**RES0**
[12]	MV	Miscellaneous syndrome. The value written to ext-ERR<n>STATUS.MV when an injected error is recorded. **0b1** Reserved, **RAO/WI**. ext-ERR<n>STATUS.MV is set to 1 when an injected error is recorded.	x
[11]	AV	Address syndrome. The value written to ext-ERR<n>STATUS.AV when an injected error is recorded. **0b0** **RES0**. An address will not be generated by the fault generation feature of the node.	x
[10]	PN	Poison flag. The value written to ext-ERR<n>STATUS.PN when an injected error is recorded. **0b0** **RES0**. ext-ERR<n>STATUS.PN is set to 0 when an injected error is recorded.	x
[9]	ER	Error Reported flag. The value written to ext-ERR<n>STATUS.ER when an injected error is recorded. **0b0** **RES0**. ext-ERR<n>STATUS.ER is set to 0 when an injected error is recorded.	x
[8]	CI	Critical Error flag. The value written to ext-ERR<n>STATUS.CI when an injected error is recorded. **0b0** **RES0**. A critical error condition will not be generated by the fault generation feature of the node.	x

Bits	Name	Description	Reset
[7:6]	CE	Corrected Error generation enable. Controls the type of injected Corrected error generated by the fault injection feature of the node. **0b00** An injected Corrected error will not be generated by the fault injection feature of the node. This value (**RES0**) is reported if CPU_CACHE_PROTECTION is disabled **0b01** An injected non-specific Corrected error is generated in the fault injection state. ext-ERR<n>STATUS.CE is set to 0b10 when the injected error is recorded. **0b10** Reserved **0b11** Reserved	xx
[5]	DE	Deferred Error generation enable. Controls whether an injected Deferred error is generated by the fault injection feature of the node. **0b0** An injected Deferred error will not be generated by the fault injection feature of the node. This value (**RES0**) is reported if CPU_CACHE_PROTECTION is disabled **0b1** An injected Deferred error is generated in the fault injection state.	x
[4]	UEO	Latent or Restartable Error generation enable. Controls whether an injected Latent or Restartable error is generated by the fault injection feature of the node. **0b0** **RES0**. An injected Latent or Restartable error will not be generated by the fault generation feature of the node.	x
[3]	UER	Signaled or Recoverable Error generation enable. Controls whether an injected Signaled or Recoverable error is generated by the fault injection feature of the node. **0b0** **RES0**. An injected Signaled or Recoverable error will not be generated by the fault generation feature of the node.	x
[2]	UEU	Unrecoverable Error generation enable. Controls whether an injected Unrecoverable error is generated by the fault injection feature of the node. **0b0** **RES0**. An injected Unrecoverable error will not be generated by the fault generation feature of the node.	x
[1]	UC	Uncontainable Error generation enable. Controls whether an injected Uncontainable error is generated by the fault injection feature of the node. **0b0** An injected Uncontainable error will not be generated by the fault injection feature of the node. This value (**RES0**) is reported if CPU_CACHE_PROTECTION is disabled **0b1** An injected Uncontainable error is generated in the fault injection state.	x
[0]	OF	Overflow flag. The value written to ext-ERR<n>STATUS.OF when an injected error is recorded. **0b0** **RES0**. ext-ERR<n>STATUS.OF is set to 0 when an injected error is recorded.	x

Accessibility

Component	Offset	Instance	Range
Complex RAS	0x808	ERR0PFGCTL	None

This interface is accessible as follows:

RW

B.2.10 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

Complex RAS

Register offset

0xE00

Access type

RO

Reset value

xxxx xxxx 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 000x

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-12: complex_ras.errgsr bit assignments

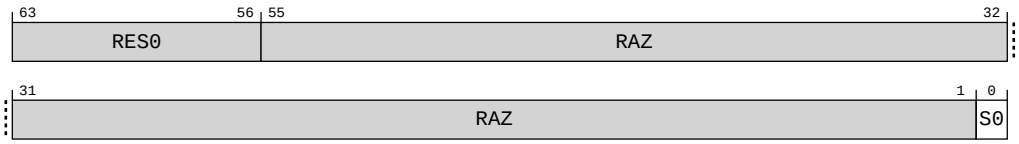


Table B-25: ERRGSR bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[55:1]	**RAZ**	Reserved	**RAZ**
[0]	S0	The status for error record <m>. A read-only copy of ERR<m>STATUS.V. **0b0** No error. **0b1** One or more errors.	x

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xE00	ERRGSR	None

This interface is accessible as follows:

RO

B.2.11 ERRIIDR, Implementation Identification Register

Defines the implementer of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Complex RAS

Register offset

0xE10

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-13: complex_ras.erriidr bit assignments

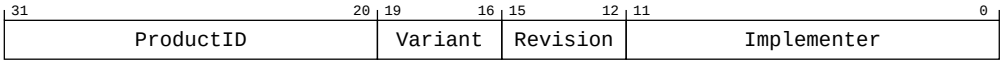


Table B-27: ERRIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	Part number, bits [11:0]. The part number is selected by the designer of the component. If ext-ERRPIDR0 and ext-ERRPIDR1 are implemented, ext-ERRPIDR0.PART_0 matches bits [7:0] of ERRIIDR.ProductID and ext-ERRPIDR1.PART_1 matches bits [11:8] of ERRIIDR.ProductID.	`12{x}`
[19:16]	Variant	Component major revision. This field distinguishes product variants or major revisions of the product. If ext-ERRPIDR2 is implemented, ext-ERRPIDR2.REVISION matches ERRIIDR.Variant.	`xxxx`
[15:12]	Revision	Component minor revision. This field distinguishes minor revisions of the product. If ext-ERRPIDR3 is implemented, ext-ERRPIDR3.REVAND matches ERRIIDR.Revision.	`xxxx`
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the RAS component. For an Arm implementation, this field has the value `0x43B`. Bits [11:8] contain the JEP106 continuation code of the implementer, and bits [6:0] contain the JEP106 identity code of the implementer. Bit 7 is **RES0**. If ext-ERRPIDR4 is implemented, ext-ERRPIDR2 is implemented, and ext-ERRPIDR1 is implemented, ext-ERRPIDR4.DES_2 matches bits [11:8] of ERRIIDR.Implementer, ext-ERRPIDR2.DES_1 matches bits [6:4] of ERRIIDR.Implementer, and ext-ERRPIDR1.DES_0 matches bits [3:0] of ERRIIDR.Implementer.	`12{x}`

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xE10	ERRIIDR	None

This interface is accessible as follows:

RO

B.2.12 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 AArch64-MPIDR_EL1 or part of AArch64-MPIDR_EL1:

- If the group of error records has affinity with a single PE, the affinity level is 0, ERRDEVAFF reads the same value as AArch64-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, parts of ERRDEVAFF reads the same value as parts of AArch64-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 AArch64-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 AArch64-MPIDR_EL1.Aff1[7:3] == ERRDEVAFF.Aff1[7:3].

If RAS System Architecture v1.1 is not implemented, 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.

Attributes

Width

64

Component

Complex RAS

Register offset

0xFA8

Access type

RO

Reset value

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

Figure B-14: complex_ras.errdevaff bit assignments

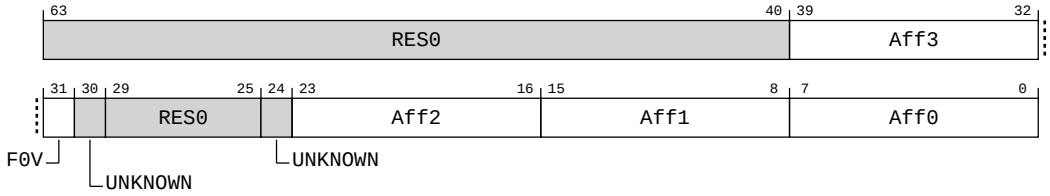


Table B-29: ERRDEVAFF bit descriptions

Bits	Name	Description	Reset
[63:40]	**RES0**	Reserved	**RES0**
[39:32]	Aff3	PE affinity level 3. The AArch64-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. **0b0** ERRDEVAFF.Aff0 is not valid, and the PE affinity is above level 0 or a subset of level 0.	x
[30]	**UNKNOWN**	Reserved	**UNKNOWN**
[29:25]	**RES0**	Reserved	**RES0**
[24]	**UNKNOWN**	Reserved	**UNKNOWN**
[23:16]	Aff2	PE affinity level 2. The AArch64-MPIDR_EL1.Aff2 field, viewed from the highest Exception level of the associated PE or PEs.	8{x}
[15:8]	Aff1	PE affinity level 1. Defines part of the AArch64-MPIDR_EL1.Aff1 field, viewed from the highest Exception level of the associated PEs. **xxxxxxx1** ERRDEVAFF.Aff1[7:1] is the value of AArch64-MPIDR_EL1.Aff1[7:1], viewed from the highest Exception level of the associated PEs.	8{x}
[7:0]	Aff0	PE affinity level 0. Indicates whether the PE affinity is at level 1. **0b00000000** PE affinity is above level 1 or a subset of level 1.	8{x}

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xFA8	ERRDEVAFF	None

This interface is accessible as follows:

RO

B.2.13 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

Complex RAS

Register offset

0xFBC

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-15: complex_ras.errdevarch bit assignments

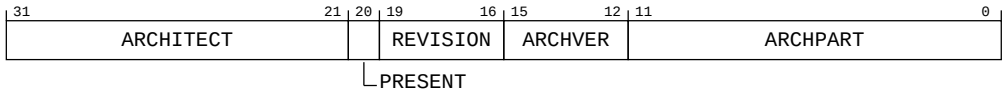


Table B-31: ERRDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code. **0b01000111011** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited. Other values are defined by the JEDEC JEP106 standard. This field reads as 0x23B.	11{x}

Bits	Name	Description	Reset
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. **0b0** Device Architecture information not present. **0b1** Device Architecture information present. This field reads as 1.	x
[19:16]	REVISION	Revision. Defines the architecture revision of the component. **0b0000** RAS System Architecture v1.0. **0b0001** RAS System Architecture v1.1. As 0b0000 and also: • Simplifies ext-ERR<n>STATUS. • Adds support for additional ERR<n>MISC<m> registers. • Adds support for the optional RAS Timestamp Extension. • Adds support for the optional Common Fault Injection Model Extension. All other values are reserved.	xxxx
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **0b0000** RAS System Architecture v1. All other values are reserved. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12]. This field reads as 0b0000.	xxxx
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **0b101000000000** RAS System Architecture. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHPART is ARCHID[11:0]. This field reads as 0xA00.	12{x}

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xFBC	ERRDEVARCH	None

This interface is accessible as follows:

RO

B.2.14 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

Complex RAS

Register offset

0xFC8

Access type

RO

Reset value

xxxx xxxx xxxx xxxx 0000 0000 0000 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-16: complex_ras.errdevid bit assignments

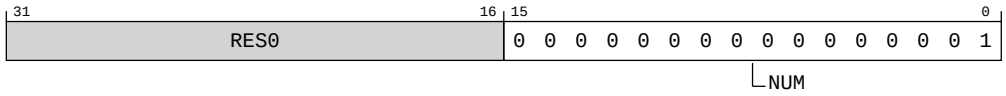


Table B-33: ERRDEVID bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[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. **0b0000000000000001** One record present.	0x0001

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xFC8	ERRDEVID	None

This interface is accessible as follows:

RO

B.2.15 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

Complex RAS

Register offset

0xFD0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-17: complex_ras.errpidr4 bit assignments

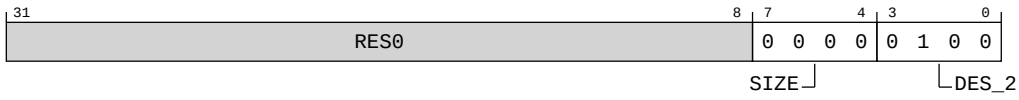


Table B-35: ERRPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	Size of the component. The distance from the start of the address space used by this component to the end of the component identification registers. A value of `0b0000` means one of the following is true: • The component uses a single 4KB block. • The component uses an **IMPLEMENTATION DEFINED** number of 4KB blocks. Any other value means the component occupies $2^{ERRPIDR4.SIZE}$ 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
Complex RAS	0xFD0	ERRPIDR4	None

This interface is accessible as follows:

RO

B.2.16 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

Complex RAS

Register offset

0xFE0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1000 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-18: complex_ras.errpidr0 bit assignments

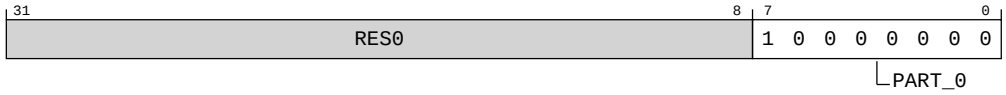


Table B-37: ERRPIDR0 bit descriptions

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, it is stored in ext-ERRPIDR1.PART_1 and ERRPIDR0.PART_0. There are 8 bits, ext-ERRPIDR2.REVISION and ext-ERRPIDR3.REVAND, available to define the revision of the component. • If a 16-bit part number is used, it is stored in ext-ERRPIDR2.PART_2, ext-ERRPIDR1.PART_1 and ERRPIDR0.PART_0. There are 4 bits, ext-ERRPIDR3.REVISION, available to define the revision of the component. **0b10000000** Cortex-A520	0x80

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xFE0	ERRPIDR0	None

This interface is accessible as follows:

RO

B.2.17 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

Complex RAS

Register offset

0xFE4

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-19: complex_ras.errpidr1 bit assignments

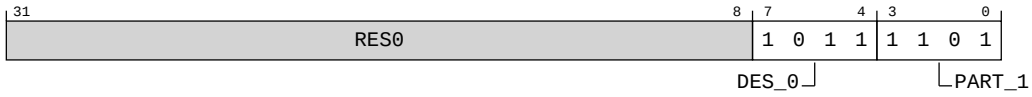


Table B-39: ERRPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7:4]	DES_0	Designer, JEP106 identification code, bits [3:0]. ERRPIDR1.DES_0 and ext-ERRPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. **0b1011** Arm Limited	0b1011
[3:0]	PART_1	Part number, bits [11:8]. The part number is selected by the designer of the component. The designer chooses whether to use a 12-bit or a 16-bit part number: • If a 12-bit part number is used, it is stored in ERRPIDR1.PART_1 and ext-ERRPIDR0.PART_0. There are 8 bits, ext-ERRPIDR2.REVISION and ext-ERRPIDR3.REVAND, available to define the revision of the component. • If a 16-bit part number is used, it is stored in ext-ERRPIDR2.PART_2, ERRPIDR1.PART_1 and ext-ERRPIDR0.PART_0. There are 4 bits, ext-ERRPIDR3.REVISION, available to define the revision of the component. **0b1101** Cortex-A520	0b1101

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xFE4	ERRPIDR1	None

This interface is accessible as follows:

RO

B.2.18 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

Complex RAS

Register offset

0xFE8

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-20: complex_ras.errpidr2 bit assignments

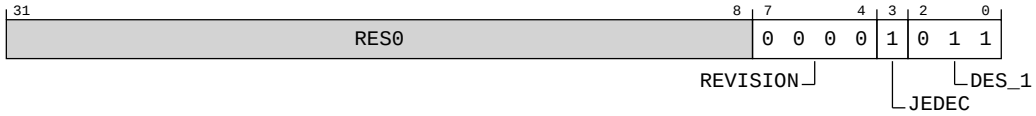


Table B-41: ERRPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Component major revision. ERRPIDR2.REVISION and ext-ERRPIDR3.REVAND together form the revision number of the component, with ERRPIDR2.REVISION being the most significant part and ext-ERRPIDR3.REVAND the least significant part. When a component is changed, ERRPIDR2.REVISION or ext-ERRPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. ext-ERRPIDR3.REVAND should be set to `0b0000` when ERRPIDR2.REVISION is increased. **`0b0000`** r0p2	`0b0000`
[3]	JEDEC	JEDEC-assigned JEP106 implementer code is used. **`0b1`**	`0b1`
[2:0]	DES_1	Designer, JEP106 identification code, bits [6:4]. ext-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
Complex RAS	0xFE8	ERRPIDR2	None

This interface is accessible as follows:

RO

B.2.19 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

Complex RAS

Register offset

0xFEC

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0010 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-21: complex_ras.errpidr3 bit assignments

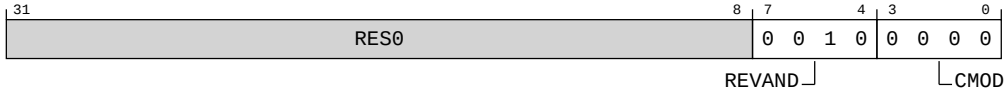


Table B-43: ERRPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7:4]	REVAND	Component minor revision. ext-ERRPIDR2.REVISION and ERRPIDR3.REVAND together form the revision number of the component, with ext-ERRPIDR2.REVISION being the most significant part and ERRPIDR3.REVAND the least significant part. When a component is changed, ext-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 ext-ERRPIDR2.REVISION is increased. **`0b0010`** r0p2	`0b0010`
[3:0]	CMOD	Customer Modified. Indicates the component has been modified. A value of `0b0000` means the component is not modified from the original design. Any other value means the component has been modified in an **IMPLEMENTATION DEFINED** way. **`0b0000`**	`0b0000`

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xFEC	ERRPIDR3	None

This interface is accessible as follows:

RO

B.2.20 ERRCIDR0, Component 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

ERRCIDR0 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

Complex RAS

Register offset

0xFF0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-22: complex_ras.errcidr0 bit assignments

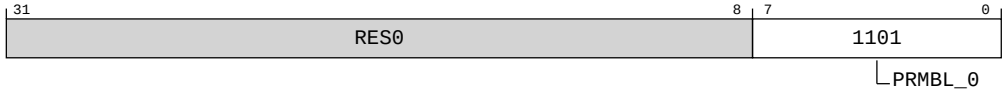


Table B-45: ERRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Component identification preamble, segment 0. **0b00001101**	0x0D

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xFF0	ERRCIDR0	None

This interface is accessible as follows:

RO

B.2.21 ERRCIDR1, Component 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

ERRCIDR1 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

Complex RAS

Register offset

0xFF4

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-23: complex_ras.errcidr1 bit assignments

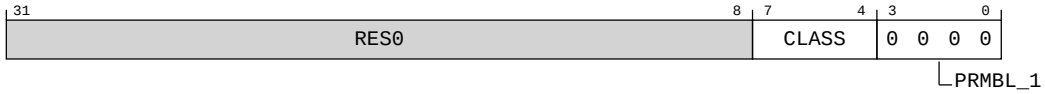


Table B-47: ERRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1111** Generic peripheral with **IMPLEMENTATION DEFINED** register layout. Other values are defined by the CoreSight Architecture. This field reads as `0xF`.	xxxx
[3:0]	PRMBL_1	Component identification preamble, segment 1. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xFF4	ERRCIDR1	None

This interface is accessible as follows:

RO

B.2.22 ERRCIDR2, Component 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

ERRCIDR2 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

Complex RAS

Register offset

0xFF8

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-24: complex_ras.errcidr2 bit assignments

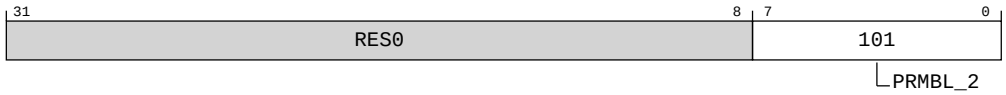


Table B-49: ERRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Component identification preamble, segment 2. 0b00000101	0x05

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xFF8	ERRCIDR2	None

This interface is accessible as follows:

RO

B.2.23 ERRCIDR3, Component 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

ERRCIDR3 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

Complex RAS

Register offset

0xFFC

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-25: complex_ras.errcidr3 bit assignments

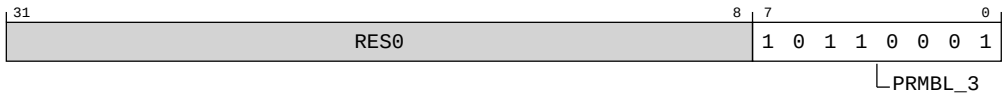


Table B-51: ERRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Component identification preamble, segment 3. **0b10110001**	0xB1

Accessibility

Component	Offset	Instance	Range
Complex RAS	0xFFC	ERRCIDR3	None

This interface is accessible as follows:

RO

B.3 External Core RAS registers summary

The summary table provides an overview of all memory-mapped Core 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 B-53: Core RAS registers summary

Offset	Name	Reset	Width	Description
0x0	ERR0FR	—	64-bit	Error Record Feature Register
0x8	ERR0CTLR	—	64-bit	Error Record Control Register
0x10	ERR0STATUS	—	64-bit	Error Record Primary Status Register
0x20	ERR0MISC0	—	64-bit	Error Record Miscellaneous Register 0
0x28	ERR0MISC1	—	64-bit	Error Record Miscellaneous Register 1
0x30	ERR0MISC2	—	64-bit	Error Record Miscellaneous Register 2
0x38	ERR0MISC3	—	64-bit	Error Record Miscellaneous Register 3
0x800	ERR0PFGF	—	64-bit	Pseudo-fault Generation Feature Register
0x808	ERR0PFGCTL	—	64-bit	Pseudo-fault Generation Control Register
0x810	ERR0PFGCDN	—	64-bit	Pseudo-fault Generation Countdown Register
0xE00	ERRGSR	—	64-bit	Error Group Status Register
0xE10	ERRIIDR	—	32-bit	Implementation Identification Register
0xFA8	ERRDEVAFF	—	64-bit	Device Affinity Register

Offset	Name	Reset	Width	Description
0xFBC	ERRDEVARCH	–	32-bit	Device Architecture Register
0xFC8	ERRDEVID	–	32-bit	Device Configuration Register
0xFD0	ERRPIDR4	–	32-bit	Peripheral Identification Register 4
0xFE0	ERRPIDR0	–	32-bit	Peripheral Identification Register 0
0xFE4	ERRPIDR1	–	32-bit	Peripheral Identification Register 1
0xFE8	ERRPIDR2	–	32-bit	Peripheral Identification Register 2
0xFEC	ERRPIDR3	–	32-bit	Peripheral Identification Register 3
0xFF0	ERRCIDR0	–	32-bit	Component Identification Register 0
0xFF4	ERRCIDR1	–	32-bit	Component Identification Register 1
0xFF8	ERRCIDR2	–	32-bit	Component Identification Register 2
0xFFC	ERRCIDR3	–	32-bit	Component Identification Register 3

B.3.1 ERR0FR, Error Record Feature Register

Defines whether <n> is the first record owned by a node:

- If <n> is the first error record owned by a node, then ERR<n>FR.ED is not 0b00.
- If <n> is not the first error record owned by a node, then ERR<n>FR.ED is 0b00.

If <n> is the first record owned by the node, defines which of the common architecturally-defined features are implemented by the node and, of the implemented features, which are software programmable.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Core RAS

Register offset

0x0

Access type

RO

Reset value

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

Figure B-26: core_ras.err0fr bit assignments

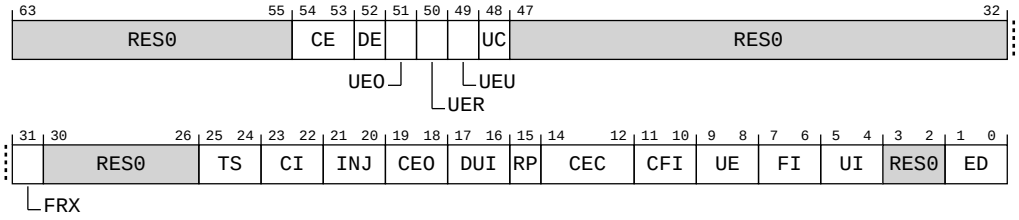


Table B-54: ERR0FR bit descriptions

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. **0b00** Does not record Corrected Errors. This value is reported if CPU_CACHE_PROTECTION is disabled **0b10** Records non-specific Corrected Errors. This value is reported if CPU_CACHE_PROTECTION is enabled	xx
[52]	DE	Deferred Error recording. Describes whether the node supports recording Deferred errors. **0b1** Records Deferred errors.	x
[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.	x
[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.	x
[49]	UEU	Unrecoverable Error recording. Describes whether the node supports recording Unrecoverable errors. **0b1** Records Unrecoverable errors.	x
[48]	UC	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. **0b1** Records Uncontainable errors.	x
[47:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	FRX	Feature Register extension. Defines whether ERR<n>FR[63:48] are architecturally defined. **0b1** ERR<n>FR[63:48] are defined by the architecture.	x
[30:26]	**RES0**	Reserved	**RES0**
[25:24]	TS	Timestamp Extension. Indicates whether, for each error record <m> owned by this node, ERR<m>MISC3 is used as the timestamp register, and, if it is, the timebase used by the timestamp. **0b00** Does not support a timestamp register.	xx
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented by the node. **0b00** Does not support the critical error interrupt. ext-ERR<n>CTLR.CI is **RES0**.	xx
[21:20]	INJ	Fault Injection Extension. Indicates whether the Common Fault Injection Model Extension is implemented by the node. **0b01** Supports the Common Fault Injection Model Extension. See ext-ERR<n>PFGF for more information.	xx
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior of the node when a second or subsequent Corrected error is recorded and a first Corrected error has previously been recorded by an error record <m> owned by the node. **0b00** Keeps the previous error syndrome.	xx
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the enabling and disabling of error recovery interrupts on deferred errors is supported by the node. **0b10** Enabling and disabling of error recovery interrupts on deferred errors is supported and controllable using ext-ERR<n>CTLR.DUI.	xx
[15]	RP	Repeat counter. Indicates whether the node implements a second Corrected error counter in ERR<m>MISC0 for each error record <m> owned by the node that can record countable errors. **0b1** Implements a first (repeat) counter and a second (other) counter in ERR<m>MISC0 for each error record <m> owned by the node that can record countable errors. The repeat counter is the same size as the primary error counter.	x
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter mechanisms in ERR<m>MISC0 for each error record <m> owned by the node that can record countable errors. **0b010** Implements an 8-bit Corrected error counter in ERR<m>MISC0[39:32] for each error record <m> owned by the node that can record countable errors.	xxx
[11:10]	CFI	Fault handling interrupt for corrected errors control. Indicates whether the enabling and disabling of fault handling interrupts on corrected errors is supported by the node. **0b10** Enabling and disabling of fault handling interrupts on corrected errors is supported and controllable using ext-ERR<n>CTLR.CFI.	xx
[9:8]	UE	In-band error reponse (External Abort). Indicates whether the in-band error response and associated controls are implemented by the node. **0b01** In-band error response is supported and always enabled. ext-ERR<n>CTLR.UE is **RES0**.	xx

Bits	Name	Description	Reset
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented by the node. **0b10** Fault handling interrupt is supported and controllable using ext-ERR<n>CTLR.FI.	xx
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented by the node. **0b10** Error handling interrupt is supported and controllable using ext-ERR<n>CTLR.UI.	xx
[3:2]	**RES0**	Reserved	**RES0**
[1:0]	ED	Error reporting and logging. Indicates error record <n> is the first record owned the node, and whether the node implements the controls for enabling and disabling error reporting and logging. **0b10** Error reporting and logging is controllable using ext-ERR<n>CTLR.ED.	xx

Accessibility

Component	Offset	Instance	Range
Core RAS	0x0	ERR0FR	None

This interface is accessible as follows:

RO

B.3.2 ERR0CTLR, Error Record Control Register

The error control register contains enable bits for the node that writes to this record:

- Enabling error detection and correction.
- Enabling the critical error, error recovery, and fault handling interrupts.
- Enabling in-band error response for uncorrected errors.

For each bit, if the node does not support the feature, then the bit is **RES0**. The definition of each record is IMPLEMENTATION DEFINED.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Core RAS

Register offset

0x8

Access type

RW

Reset value

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

Figure B-27: core_ras.err0ctlr bit assignments

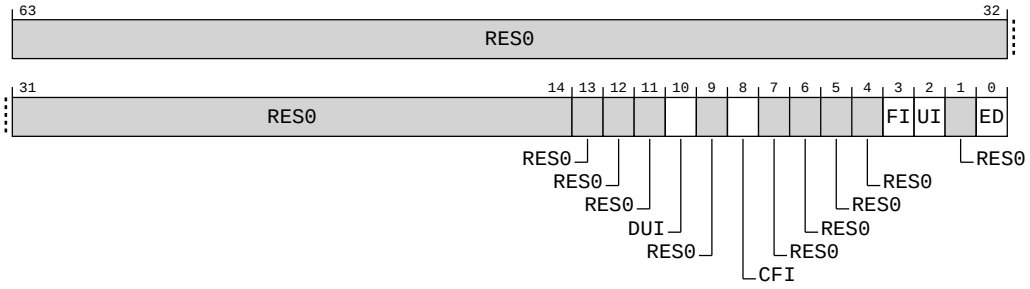


Table B-56: ERR0CTLR bit descriptions

Bits	Name	Description	Reset
[63:11]	**RES0**	Reserved	**RES0**
[10]	DUI	Error recovery interrupt for Deferred errors enable. When ext-ERR<n>FR.DUI == 0b10, this control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all errors recorded as Deferred error. **0b0** Error recovery interrupt not generated for Deferred errors. **0b1** Error recovery interrupt generated for Deferred errors. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	x
[9]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[8]	CFI	Fault handling interrupt for Corrected errors enable. When ext-ERR<n>FR.CFI == `0b10`, this control applies to errors arising from both reads and writes. When enabled: • If the node implements Corrected error counters, then the fault handling interrupt is generated when a counter overflows and the overflow bit for the counter is set to 1. For more information, see ext-ERR<n>MISC0. • Otherwise, the fault handling interrupt is also generated for all errors recorded as Corrected error. **0b0** Fault handling interrupt not generated for Corrected errors. **0b1** Fault handling interrupt generated for Corrected errors. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	x
[7:4]	**RES0**	Reserved	**RES0**
[3]	FI	Fault handling interrupt enable. When ext-ERR<n>FR.FI == `0b10`, this control applies to errors arising from both reads and writes. When enabled: • The fault handling interrupt is generated for all errors recorded as either Deferred error or Uncorrected error. • If the fault handling interrupt for Corrected errors control is not implemented: ◦ If the node implements Corrected error counters, then the fault handling interrupt is also generated when a counter overflows and the overflow bit for the counter is set to 1. ◦ Otherwise, the fault handling interrupt is also generated for all errors recorded as Corrected error. **0b0** Fault handling interrupt disabled. **0b1** Fault handling interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	x
[2]	UI	Uncorrected error recovery interrupt enable. When ext-ERR<n>FR.UI == `0b10`, this control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all errors recorded as Uncorrected error. **0b0** Error recovery interrupt disabled. **0b1** Error recovery interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	x
[1]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[0]	ED	Error reporting and logging enable. When disabled, the node behaves as if error detection and correction are disabled, and no errors are recorded or signaled by the node. Arm recommends that, when disabled, correct error detection and correction codes are written for writes, unless disabled by an **IMPLEMENTATION DEFINED** control for error injection. **0b0** Error reporting disabled. **0b1** Error reporting enabled. It is IMPLEMENTATION DEFINED whether the node fully disables error detection and correction when reporting is disabled. That is, even with error reporting disabled, the node might continue to silently correct errors. Uncorrected errors might result in corrupt data being silently propagated by the node. **Note:** If this node requires initialization after Cold reset to prevent signaling false errors, then Arm recommends this field is set to 0 on Cold reset, meaning errors are not reported from Cold reset. This allows boot software to initialize a node without signaling errors. Software can enable error reporting after the node is initialized. Otherwise, the Cold reset value is IMPLEMENTATION DEFINED. If the Cold reset value is 1, the reset values of other controls in this register are also IMPLEMENTATION DEFINED and should not be **UNKNOWN**.	x

Accessibility

Component	Offset	Instance	Range
Core RAS	0x8	ERR0CTLR	None

This interface is accessible as follows:

RW

B.3.3 ERR0STATUS, Error Record Primary Status Register

Contains status information for error record <n>, 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:

- ERR<n>STATUS.{AV, V, MV} are valid bits that define whether error record <n> registers are valid.
- ERR<n>STATUS.{UE, OF, CE, DE, UET} encode the types of error or errors recorded.
- ERR<n>STATUS.{CI, ER, PN, IERR, SERR} are syndrome fields.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Core RAS

Register offset

0x10

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x0xx x0xx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-28: core_ras.err0status bit assignments

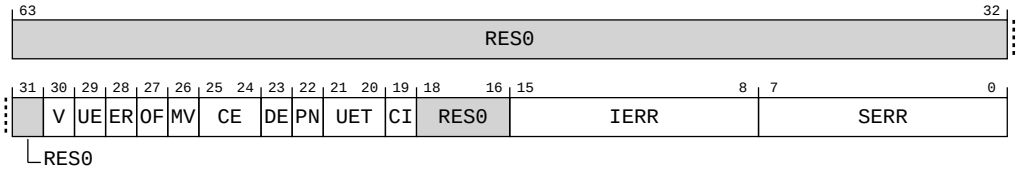


Table B-58: ERR0STATUS bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[30]	V	Status Register Valid. **0b0** ERR<n>STATUS not valid. **0b1** ERR<n>STATUS valid. At least one error has been recorded. Access to this field is: W1C	0b0
[29]	UE	Uncorrected Error. **0b0** No errors have been detected, or all detected errors have been either corrected or deferred. **0b1** At least one detected error was not corrected and not deferred. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x

Bits	Name	Description	Reset
[28]	ER	Error Reported. **0b0** No in-band error response (External Abort) signaled to the Requester making the access or other transaction. **0b1** An in-band error response was signaled by the component to the Requester making the access or other transaction. This can be because any of the following are true: • The applicable one of the ERR<q>CTLR.{WUE, RUE, UE} fields is implemented and was 1 when an error was detected and not corrected. • The applicable one of the ERR<q>CTLR.{WUE, RUE, UE} fields is not implemented and the component always reports errors. It is **IMPLEMENTATION DEFINED** whether an uncorrected error that is deferred and recorded as a Deferred error, but is not deferred to the Requester, will signal an in-band error response to the Requester, causing this field to be set to 1. If no in-band error response to the Requester, this field is set to 0. **Note:** An in-band error response signaled by the component might be masked and not generate any exception. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.UE == '0' && ext-ERR0STATUS.DE == '0' && this field can be set to 0b1 by a Deferred error** Access to this field is: **UNKNOWN**/WI **When ext-ERR0STATUS.UE == '0' && this field is never set to 0b1 by a Deferred error** Access to this field is: **UNKNOWN**/WI **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x

Bits	Name	Description	Reset
[27]	OF	Overflow. Indicates that multiple errors have been detected. This field is set to 1 when one of the following occurs: • A Corrected error counter is implemented, an error is counted, and the counter overflows. • ERR<n>STATUS.V was previously 1, a Corrected error counter is not implemented, and a Corrected error is recorded. • ERR<n>STATUS.V was previously 1, and a type of error other than a Corrected error is recorded. Otherwise, this field is unchanged when an error is recorded. If a Corrected error counter is implemented: • A direct write that modifies the counter overflow flag indirectly might set this field to an **UNKNOWN** value. • A direct write to this field that clears this field to zero might indirectly set the counter overflow flag to an **UNKNOWN** value. **0b0** Since this field was last cleared to zero, no error syndrome has been discarded and, if a Corrected error counter is implemented, it has not overflowed. **0b1** Since this field was last cleared to zero, at least one error syndrome has been discarded or, if a Corrected error counter is implemented, it might have overflowed. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x
[26]	MV	Miscellaneous Registers Valid. **0b0** ERR<n>MISC<m> not valid. **0b1** The **IMPLEMENTATION DEFINED** contents of the ERR<n>MISC<m> registers contains additional information for an error recorded by this record. **Note:** If the ERR<n>MISC<m> registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded. Access to this field is: W1C	0b0

Bits	Name	Description	Reset
[25:24]	CE	Corrected Error. **0b00** No errors were corrected. **0b10** At least one error was corrected. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	xx
[23]	DE	Deferred Error. **0b0** No errors were deferred. **0b1** At least one error was not corrected and deferred. Support for deferring errors is **IMPLEMENTATION DEFINED**. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x
[22]	PN	Poison. **0b0** Uncorrected error or Deferred error recorded because a corrupt value was detected, for example, by an error detection code (EDC), or Corrected error recorded. **0b1** Uncorrected error or Deferred error recorded because a poison value was detected. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0' || (ext-ERR0STATUS.DE == '0' && ext-ERR0STATUS.UE == '0')** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x

Bits	Name	Description	Reset
[21:20]	UET	Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. **0b00** Uncorrected error, Uncontainable error (UC). **0b01** Uncorrected error, Unrecoverable error (UEU). **0b10** Uncorrected error, Latent or Restartable error (UEO). **0b11** Uncorrected error, Signaled or Recoverable error (UER). **Note:** Software might use the information in the error record registers to determine what recovery is necessary. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. **When ext-ERR0STATUS.V == '0' || ext-ERR0STATUS.UE == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	xx
[19]	CI	Critical Error. Indicates whether a critical error condition has been recorded. **0b0** No critical error condition. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x
[18:16]	**RES0**	Reserved	**RES0**
[15:8]	IERR	**IMPLEMENTATION DEFINED** error code. Used with any primary error code ERR<n>STATUS.SERR value. Further **IMPLEMENTATION DEFINED** information can be placed in the ERR<n>MISC<m> registers. The implemented set of valid values that this field can take is IMPLEMENTATION DEFINED. If any value not in this set is written to this register, then the value read back from this field is **UNKNOWN**. **Note:** This means that one or more bits of this field might be implemented as fixed read-as-zero or read-as-one values. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RW	8{x}

Bits	Name	Description	Reset
[7:0]	SERR	Architecturally-defined primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry. **0b00000110** Data value from associative memory. For example, ECC error on cache data. **0b00000111** Address/control value from associative memory. For example, ECC error on cache tag. **0b00001000** Data value from a TLB. For example, ECC error on TLB data. **0b00001100** Data value from (non-associative) external memory. For example, ECC error in SDRAM. **0b00010010** Error response from Completer of access. For example, error response from cache write-back. **0b00010101** Deferred error from Completer not supported at Requester. For example, poisoned data received from the Completer of an access by a Requester that cannot defer the error further. **When ext-ERR0STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RW	8{x}

Access

ERR<n>STATUS.{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. ERR<n>STATUS.{IERR, SERR} are read/write (RW) fields, although the set of implemented valid values is **IMPLEMENTATION DEFINED**. See also ext-ERR<n>PFGF.SYN.

After reading ERR<n>STATUS, 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 ERR<n>STATUS.{UE, DE, CE} are ignored if ERR<n>STATUS.OF is 1 and is not being cleared to 0.
- Writes to ERR<n>STATUS.V are ignored if any of ERR<n>STATUS.{UE, DE, CE} are nonzero and are not being cleared to zero.
- Writes to ERR<n>STATUS.{AV, MV} and the ERR<n>STATUS.{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 ERR<n>STATUS.UE, ERR<n>STATUS.DE, and ERR<n>STATUS.CE.

When RAS System Architecture v1.1 is implemented, a write to the register is ignored if all of:

- Any of ERR<n>STATUS.{V, UE, OF, CE, DE} are nonzero before the write.

- The write does not clear the nonzero ERR<n>STATUS.{V, UE, OF, CE, DE} fields to zero by writing ones to the applicable field or fields.

Some of the fields in ERR<n>STATUS are also defined as **UNKNOWN** where certain combinations of ERR<n>STATUS.{V, DE, UE} are zero. The rules for writes to ERR<n>STATUS 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 ERR<n>STATUS when ERR<n>STATUS.V is 1 results in either ERR<n>STATUS.V field being cleared to zero, or ERR<n>STATUS.V not changing. Since all fields in ERR<n>STATUS, other than ERR<n>STATUS.{AV, V, MV}, usually read as **UNKNOWN** values when ERR<n>STATUS.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:

- Read ERR<n>STATUS and determine which fields need to be cleared to zero.
- 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.

Otherwise, these fields might not have the correct value when a new fault is recorded.

An exception is when the node supports writing to these fields as part of fault injection. See also ext-ERR<n>PFGF.SYN.

Accessibility

ERR<n>STATUS.{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. ERR<n>STATUS.{IERR, SERR} are read/write (RW) fields, although the set of implemented valid values is IMPLEMENTATION DEFINED. See also ext-ERR<n>PFGF.SYN.

After reading ERR<n>STATUS, 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 ERR<n>STATUS.{UE, DE, CE} are ignored if ERR<n>STATUS.OF is 1 and is not being cleared to 0.
- Writes to ERR<n>STATUS.V are ignored if any of ERR<n>STATUS.{UE, DE, CE} are nonzero and are not being cleared to zero.
- Writes to ERR<n>STATUS.{AV, MV} and the ERR<n>STATUS.{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 ERR<n>STATUS.UE, ERR<n>STATUS.DE, and ERR<n>STATUS.CE.

When RAS System Architecture v1.1 is implemented, a write to the register is ignored if all of:

- Any of ERR<n>STATUS.{V, UE, OF, CE, DE} are nonzero before the write.
- The write does not clear the nonzero ERR<n>STATUS.{V, UE, OF, CE, DE} fields to zero by writing ones to the applicable field or fields.

Some of the fields in ERR<n>STATUS are also defined as UNKNOWN where certain combinations of ERR<n>STATUS.{V, DE, UE} are zero. The rules for writes to ERR<n>STATUS 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 ERR<n>STATUS when ERR<n>STATUS.V is 1 results in either ERR<n>STATUS.V field being cleared to zero, or ERR<n>STATUS.V not changing. Since all fields in ERR<n>STATUS, other than ERR<n>STATUS.{AV, V, MV}, usually read as UNKNOWN values when ERR<n>STATUS.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:

- Read ERR<n>STATUS and determine which fields need to be cleared to zero.
- 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.

Otherwise, these fields might not have the correct value when a new fault is recorded.

An exception is when the node supports writing to these fields as part of fault injection. See also ext-ERR<n>PFGF.SYN.

Component	Offset	Instance	Range
Core RAS	0x10	ERR0STATUS	None

This interface is accessible as follows:

When ext-ERR<n>STATUS.V != '0' && ERR<n>STATUS.V is not being cleared to 0b0 in the same write

RO

When ext-ERR<n>STATUS.UE != '0' && ERR<n>STATUS.UE is not being cleared to 0b0 in the same write

RO

When ext-ERR<n>STATUS.OF != '0' && ERR<n>STATUS.OF is not being cleared to 0b0 in the same write

RO

When ext-ERR<n>STATUS.CE != '00' && ERR<n>STATUS.CE is not being cleared to 0b00 in the same write

RO

When ext-ERR<n>STATUS.DE != '0' && ERR<n>STATUS.DE is not being cleared to 0b0 in the same write

RO

Otherwise

RW

B.3.4 ERR0MISC0, Error Record Miscellaneous Register 0

Contains information recording the cache line or TLB entry of a detected RAM error. Also contains the architecturally-defined Corrected error counters.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Core RAS

Register offset

0x20

Access type

See bit descriptions

Reset value

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

Figure B-29: core_ras.err0misc0 bit assignments

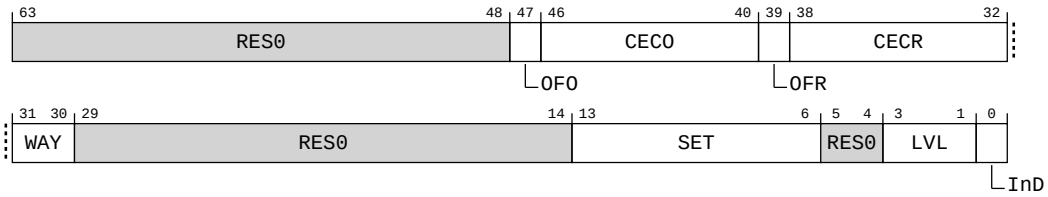


Table B-60: ERR0MISC0 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERR<n>MISC0.CECO is incremented and wraps through zero. **0b0** Other counter has not overflowed. **0b1** Other counter has overflowed. A direct write that modifies this field might indirectly set ext-ERR<n>STATUS.OF to an **UNKNOWN** value and a direct write to ext-ERR<n>STATUS.OF that clears it to zero might indirectly set this field to an **UNKNOWN** value.	x
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERR<n>MISC0.CECR.	7{x}
[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERR<n>MISC0.CECR is incremented and wraps through zero. **0b0** Repeat counter has not overflowed. **0b1** Repeat counter has overflowed. A direct write that modifies this field might indirectly set ext-ERR<n>STATUS.OF to an **UNKNOWN** value and a direct write to ext-ERR<n>STATUS.OF that clears it to zero might indirectly set this field to an **UNKNOWN** value.	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. **Note:** For example, the other syndrome might include the set and way information for an error detected in a cache. This might be recorded in the **IMPLEMENTATION DEFINED** ERR<n>MISC<m> fields on a first Corrected error. ERR<n>MISC0.CECR is then incremented for each subsequent Corrected Error in the same set and way.	7{x}
[31:30]	WAY	The way that contained the error **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	xx
[29:14]	**RES0**	Reserved	**RES0**
[13:6]	SET	The set that contained the error **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	8{x}
[5:4]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[3:1]	LVL	Cache level **0b000** L1. **0b001** L2. **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	xxx
[0]	InD	Instruction or Data cache **0b0** Data or unified cache. **0b1** Instruction cache. **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	x

Access

Reads from ERR<n>MISC0 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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.

Accessibility

Reads from ERR<n>MISC0 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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
Core RAS	0x20	ERR0MISC0	None

This interface is accessible as follows:

RW

B.3.5 ERR0MISC1, Error Record Miscellaneous Register 1

Contains information recording the exact location of a detected RAM error. Refer to the Cortex-A520 Core Configuration and Integration Manual to interpret the fields in this register.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Core RAS

Register offset

0x28

Access type

See bit descriptions

Reset value

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

Figure B-30: core_ras.err0misc1 bit assignments

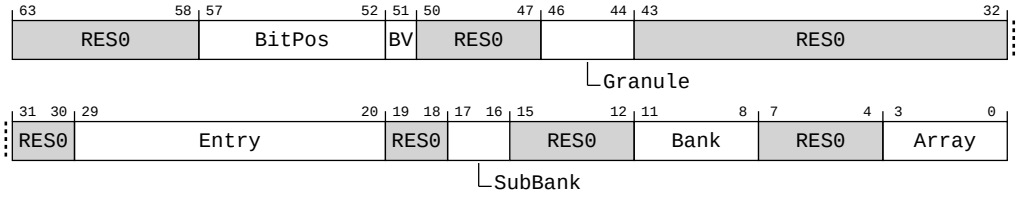


Table B-62: ERR0MISC1 bit descriptions

Bits	Name	Description	Reset
[63:58]	**RES0**	Reserved	**RES0**
[57:52]	BitPos	The bit position that contained the error **When ext-ERR0MISC1.BV == '0'** Access to this field is: **RES0** **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	6{x}
[51]	BV	Indicates that the BitPos field contains valid data **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	x
[50:47]	**RES0**	Reserved	**RES0**
[46:44]	Granule	The protection granule within the ram entry containing the error **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	xxx
[43:30]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[29:20]	Entry	The RAM row containing the error **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	10{x}
[19:18]	**RES0**	Reserved	**RES0**
[17:16]	SubBank	The sub-bank of the RAM bank containing the error **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	xx
[15:12]	**RES0**	Reserved	**RES0**
[11:8]	Bank	The RAM bank within the array containing the error **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	xxxx
[7:4]	**RES0**	Reserved	**RES0**
[3:0]	Array	The specific RAM array containing the error **0b0000** L1 D-cache data RAMs. **0b0001** L1 D-cache MTE data RAMs. **0b0010** L1 D-cache tag RAMs. **0b0011** L1 D-cache dirty RAMs. **0b0100** L1 I-cache data RAMs. **0b0101** L1 I-cache tag RAMs. **When ext-ERR0STATUS.MV == '1'** Access to this field is: RO **Otherwise** Access to this field is: RW	xxxx

Access

Reads from ERR<n>MISC1 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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.

Accessibility

Reads from ERR<n>MISC1 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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
Core RAS	0x28	ERR0MISC1	None

This interface is accessible as follows:

RW

B.3.6 ERR0MISC2, Error Record Miscellaneous Register 2

This register is not used and is reserved.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Core RAS

Register offset

0x30

Access type

Read

R

Write

W

Reset value

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

Figure B-31: core_ras.err0misc2 bit assignments

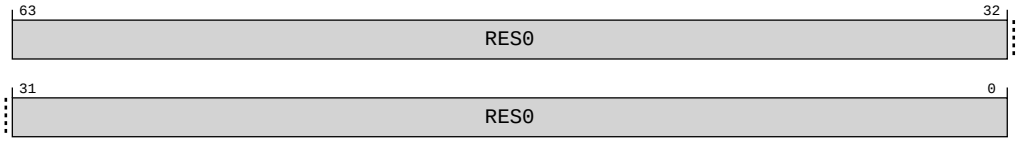


Table B-64: ERR0MISC2 bit descriptions

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

Access

Reads from ERR<n>MISC2 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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.

Accessibility

Reads from ERR<n>MISC2 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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
Core RAS	0x30	ERR0MISC2	None

This interface is accessible as follows:

RW

B.3.7 ERR0MISC3, Error Record Miscellaneous Register 3

This register is not used and is reserved.

Configurations

ERR0FR describes the features implemented by the node.

Attributes

Width

64

Component

Core RAS

Register offset

0x38

Access type

Read

R

Write

W

Reset value

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

Figure B-32: core_ras.err0misc3 bit assignments

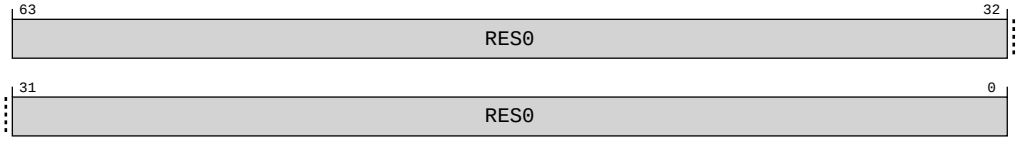


Table B-66: ERR0MISC3 bit descriptions

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

Access

Reads from ERR<n>MISC3 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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.

Accessibility

Reads from ERR<n>MISC3 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 ERR<q>PFGF.MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 1. See ext-ERR<n>PFGF.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 ext-ERR<n>STATUS.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
Core RAS	0x38	ERR0MISC3	None

This interface is accessible as follows:

RW

B.3.8 ERR0PFGF, 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

Core RAS

Register offset

0x800

Access type

RO

Reset value

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

Figure B-33: core_ras.err0pfgf bit assignments

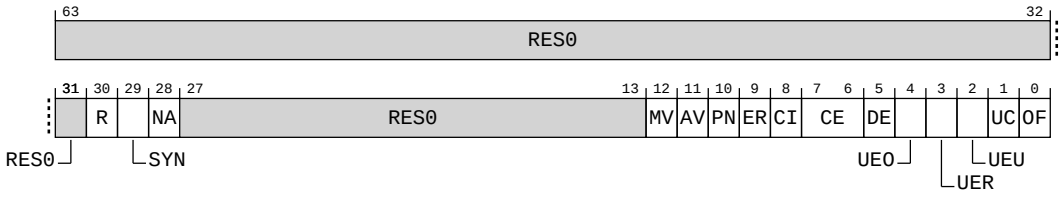


Table B-68: ERR0PFGF bit descriptions

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 ext-ERR<n>PFGCTL.R. ext-ERR<n>PFGCTL.R is a read/write field.	x

Bits	Name	Description	Reset
[29]	SYN	Syndrome. Fault syndrome injection. **0b0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.{IERR, SERR} to IMPLEMENTATION DEFINED values. ext-ERR<n>STATUS.{IERR, SERR} are **UNKNOWN** when ext-ERR<n>STATUS.V is 0.	x
[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.	x
[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. It is **IMPLEMENTATION DEFINED** which ERR<n>MISC<m> syndrome fields, if any, are updated by the node when an injected error is recorded. Some syndrome fields might always be updated by the node when an error, including an injected error, is recorded. For example, a corrected error counter might always be updated when any countable error, including a injected countable error, is recorded. **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 ext-ERR<n>STATUS.MV is set to 1. • Otherwise, ext-ERR<n>STATUS.MV is unchanged. If the node always sets ext-ERR<n>STATUS.MV to 1 when recording an injected error then ext-ERR<n>PFGCTL.MV might be RAO/WI. Otherwise ext-ERR<n>PFGCTL.MV is **RES0**.	x
[11]	AV	Address syndrome. Defines whether software can control the address recorded in ext-ERR<n>ADDR when an injected error is recorded. **0b0** When an injected error is recorded, the node might record an address in ext-ERR<n>ADDR. If an address is recorded in ext-ERR<n>ADDR, then ext-ERR<n>STATUS.AV is set to 1. Otherwise, ext-ERR<n>ADDR and ext-ERR<n>STATUS.AV are unchanged. If the node always records an address and sets ext-ERR<n>STATUS.AV to 1 when recording an injected error then ext-ERR<n>PFGCTL.AV might be RAO/WI. Otherwise ext-ERR<n>PFGCTL.AV is **RES0**.	x
[10]	PN	Poison flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.PN status flag. **0b0** When an injected error is recorded, it is IMPLEMENTATION DEFINED whether the node sets ext-ERR<n>STATUS.PN to 1. ext-ERR<n>PFGCTL.PN is **RES0**.	x
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.ER status flag. **0b0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.ER according to the architecture-defined rules for setting the ER field. ext-ERR<n>PFGCTL.ER is **RES0**.	x

Bits	Name	Description	Reset
[8]	CI	Critical Error flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.CI status flag. **0b0** When an injected error is recorded, it is IMPLEMENTATION DEFINED whether the node sets ext-ERR<n>STATUS.CI to 1. ext-ERR<n>PFGCTL.CI is **RES0**.	x
[7:6]	CE	Corrected Error generation. Describes the types of Corrected error that the fault generation feature of the node can generate. **0b00** The fault generation feature of the node does not generate Corrected errors. This value is reported if CPU_CACHE_PROTECTION is disabled **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 ERR<n>STATUS.CE == 0b10. This value is reported if CPU_CACHE_PROTECTION is enabled	xx
[5]	DE	Deferred Error generation. Describes whether the fault generation feature of the node can generate Deferred errors. **0b0** The fault generation feature of the node does not generate Deferred errors. This value is reported if CPU_CACHE_PROTECTION is disabled **0b1** The fault generation feature of the node allows generation of Deferred errors. This value is reported if CPU_CACHE_PROTECTION is enabled	x
[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. ext-ERR<n>PFGCTL.UEO is **RES0**.	x
[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. ext-ERR<n>PFGCTL.UER is **RES0**.	x
[2]	UEU	Unrecoverable Error generation. Describes whether the fault generation feature of the node can generate Unrecoverable errors. **0b1** The fault generation feature of the node allows generation of Unrecoverable errors. ext-ERR<n>PFGCTL.UEU is a read/write field.	x
[1]	UC	Uncontainable Error generation. Describes whether the fault generation feature of the node can generate Uncontainable errors. **0b0** The fault generation feature of the node does not generate Uncontainable errors. This value is reported if CPU_CACHE_PROTECTION is disabled **0b1** The fault generation feature of the node allows generation of Uncontainable errors. This value is reported if CPU_CACHE_PROTECTION is enabled	x

Bits	Name	Description	Reset
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.OF status flag. **0b0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.OF according to the architecture-defined rules for setting the OF field. ext-ERR<n>PFGCTL.OF is **RES0**.	x

Accessibility

Component	Offset	Instance	Range
Core RAS	0x800	ERR0PFGF	None

This interface is accessible as follows:

RO

B.3.9 ERR0PFGCTL, Pseudo-fault Generation Control Register

Enables controlled fault generation.

Configurations

ext-ERR<n>FR and ext-ERR<n>PFGF describe the features implemented by the node.

Attributes

Width

64

Component

Core RAS

Register offset

0x808

Access type

RW

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-34: core_ras.err0pfgctl bit assignments

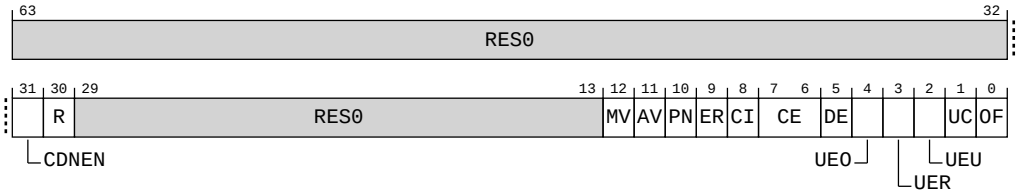


Table B-70: ERR0PFGCTL bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	CDNEN	Countdown Enable. Controls transfers of the value held in ext-ERR<n>PFGCDN to the Error Generation Counter and enables this counter. **0b0** The Error Generation Counter is disabled. **0b1** The Error Generation Counter is enabled. On a write of 1 to this field, the Error Generation Counter is set to ext-ERR<n>PFGCDN.CDN.	0b0
[30]	R	Restart. Controls whether the Error Generation Counter restarts or stops counting on reaching zero. **0b0** On reaching zero, the Error Generation Counter will stop counting. **0b1** On reaching zero, the Error Generation Counter is set to ext-ERR<n>PFGCDN.CDN.	x
[29:13]	**RES0**	Reserved	**RES0**
[12]	MV	Miscellaneous syndrome. The value written to ext-ERR<n>STATUS.MV when an injected error is recorded. **0b1** Reserved, **RAO/WI**. ext-ERR<n>STATUS.MV is set to 1 when an injected error is recorded.	x
[11]	AV	Address syndrome. The value written to ext-ERR<n>STATUS.AV when an injected error is recorded. **0b0** **RES0**. An address will not be generated by the fault generation feature of the node.	x
[10]	PN	Poison flag. The value written to ext-ERR<n>STATUS.PN when an injected error is recorded. **0b0** **RES0**. ext-ERR<n>STATUS.PN is set to 0 when an injected error is recorded.	x
[9]	ER	Error Reported flag. The value written to ext-ERR<n>STATUS.ER when an injected error is recorded. **0b0** **RES0**. ext-ERR<n>STATUS.ER is set to 0 when an injected error is recorded.	x
[8]	CI	Critical Error flag. The value written to ext-ERR<n>STATUS.CI when an injected error is recorded. **0b0** **RES0**. A critical error condition will not be generated by the fault generation feature of the node.	x

Bits	Name	Description	Reset
[7:6]	CE	Corrected Error generation enable. Controls the type of injected Corrected error generated by the fault injection feature of the node. **0b00** An injected Corrected error will not be generated by the fault injection feature of the node. This value (**RES0**) is reported if CPU_CACHE_PROTECTION is disabled **0b01** An injected non-specific Corrected error is generated in the fault injection state. ext-ERR<n>STATUS.CE is set to 0b10 when the injected error is recorded. **0b10** Reserved **0b11** Reserved	xx
[5]	DE	Deferred Error generation enable. Controls whether an injected Deferred error is generated by the fault injection feature of the node. **0b0** An injected Deferred error will not be generated by the fault injection feature of the node. This value (**RES0**) is reported if CPU_CACHE_PROTECTION is disabled **0b1** An injected Deferred error is generated in the fault injection state.	x
[4]	UEO	Latent or Restartable Error generation enable. Controls whether an injected Latent or Restartable error is generated by the fault injection feature of the node. **0b0** **RES0**. An injected Latent or Restartable error will not be generated by the fault generation feature of the node.	x
[3]	UER	Signaled or Recoverable Error generation enable. Controls whether an injected Signaled or Recoverable error is generated by the fault injection feature of the node. **0b0** **RES0**. An injected Signaled or Recoverable error will not be generated by the fault generation feature of the node.	x
[2]	UEU	Unrecoverable Error generation enable. Controls whether an injected Unrecoverable error is generated by the fault injection feature of the node. **0b0** An injected Unrecoverable error will not be generated by the fault generation feature of the node. **0b1** An injected Unrecoverable error is generated in the fault injection state. The node enters the fault injection state when the Error Generation Counter decrements to zero. It is **IMPLEMENTATION DEFINED** whether the injected error is generated when the error is generated on an access to the component in the fault injection state and the data is not consumed.	x
[1]	UC	Uncontainable Error generation enable. Controls whether an injected Uncontainable error is generated by the fault injection feature of the node. **0b0** An injected Uncontainable error will not be generated by the fault injection feature of the node. This value (**RES0**) is reported if CPU_CACHE_PROTECTION is disabled **0b1** An injected Uncontainable error is generated in the fault injection state.	x

Bits	Name	Description	Reset
[0]	OF	Overflow flag. The value written to ext-ERR<n>STATUS.OF when an injected error is recorded. **0b0** **RES0**. ext-ERR<n>STATUS.OF is set to 0 when an injected error is recorded.	x

Accessibility

Component	Offset	Instance	Range
Core RAS	0x808	ERR0PFGCTL	None

This interface is accessible as follows:

RW

B.3.10 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

Core RAS

Register offset

0xE00

Access type

RO

Reset value

xxxx xxxx 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 000x

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-35: core_ras.errgsr bit assignments

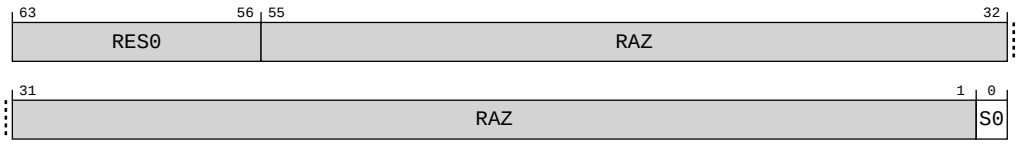


Table B-72: ERRGSR bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[55:1]	**RAZ**	Reserved	**RAZ**
[0]	S0	The status for error record <m>. A read-only copy of ERR<m>STATUS.V. **0b0** No error. **0b1** One or more errors.	x

Accessibility

Component	Offset	Instance	Range
Core RAS	0xE00	ERRGSR	None

This interface is accessible as follows:

RO

B.3.11 ERRIIDR, Implementation Identification Register

Defines the implementer of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Core RAS

Register offset

0xE10

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-36: core_ras.erriidr bit assignments

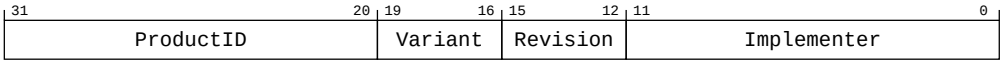


Table B-74: ERRIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	Part number, bits [11:0]. The part number is selected by the designer of the component. If ext-ERRPIDR0 and ext-ERRPIDR1 are implemented, ext-ERRPIDR0.PART_0 matches bits [7:0] of ERRIIDR.ProductID and ext-ERRPIDR1.PART_1 matches bits [11:8] of ERRIIDR.ProductID.	12{x}
[19:16]	Variant	Component major revision. This field distinguishes product variants or major revisions of the product. If ext-ERRPIDR2 is implemented, ext-ERRPIDR2.REVISION matches ERRIIDR.Variant.	xxxx
[15:12]	Revision	Component minor revision. This field distinguishes minor revisions of the product. If ext-ERRPIDR3 is implemented, ext-ERRPIDR3.REVAND matches ERRIIDR.Revision.	xxxx
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the RAS component. For an Arm implementation, this field has the value 0x43B. Bits [11:8] contain the JEP106 continuation code of the implementer, and bits [6:0] contain the JEP106 identity code of the implementer. Bit 7 is **RES0**. If ext-ERRPIDR4 is implemented, ext-ERRPIDR2 is implemented, and ext-ERRPIDR1 is implemented, ext-ERRPIDR4.DES_2 matches bits [11:8] of ERRIIDR.Implementer, ext-ERRPIDR2.DES_1 matches bits [6:4] of ERRIIDR.Implementer, and ext-ERRPIDR1.DES_0 matches bits [3:0] of ERRIIDR.Implementer.	12{x}

Accessibility

Component	Offset	Instance	Range
Core RAS	0xE10	ERRIIDR	None

This interface is accessible as follows:

RO

B.3.12 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 AArch64-MPIDR_EL1 or part of AArch64-MPIDR_EL1:

- If the group of error records has affinity with a single PE, the affinity level is 0, ERRDEVAFF reads the same value as AArch64-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, parts of ERRDEVAFF reads the same value as parts of AArch64-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 AArch64-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 AArch64-MPIDR_EL1.Aff1[7:3] == ERRDEVAFF.Aff1[7:3].

If RAS System Architecture v1.1 is not implemented, 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.

Attributes

Width

64

Component

Core RAS

Register offset

0xFA8

Access type

RO

Reset value

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

Figure B-37: core_ras.errdevaff bit assignments

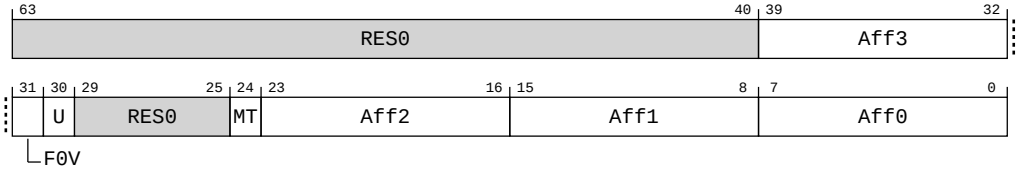


Table B-76: ERRDEVAFF bit descriptions

Bits	Name	Description	Reset
[63:40]	**RES0**	Reserved	**RES0**
[39:32]	Aff3	PE affinity level 3. The AArch64-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.	x
[30]	U	Uniprocessor. The AArch64-MPIDR_EL1.U field, viewed from the highest Exception level of the associated PE.	x
[29:25]	**RES0**	Reserved	**RES0**
[24]	MT	Multithreaded. The AArch64-MPIDR_EL1.MT field, viewed from the highest Exception level of the associated PE.	x
[23:16]	Aff2	PE affinity level 2. The AArch64-MPIDR_EL1.Aff2 field, viewed from the highest Exception level of the associated PE or PEs.	8{x}
[15:8]	Aff1	PE affinity level 1. The AArch64-MPIDR_EL1.Aff1 field, viewed from the highest Exception level of the associated PE or PEs.	8{x}
[7:0]	Aff0	PE affinity level 0. The AArch64-MPIDR_EL1.Aff0 field, viewed from the highest Exception level of the associated PE.	8{x}

Accessibility

Component	Offset	Instance	Range
Core RAS	0xFA8	ERRDEVAFF	None

This interface is accessible as follows:

RO

B.3.13 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

Core RAS

Register offset

0xFBC

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-38: core_ras.errdevarch bit assignments

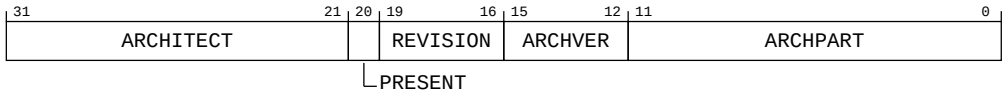


Table B-78: ERRDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code. **0b01000111011** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited. Other values are defined by the JEDEC JEP106 standard. This field reads as 0x23B.	11{x}

Bits	Name	Description	Reset
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. **0b1** Device Architecture information present.	x
[19:16]	REVISION	Revision. Defines the architecture revision of the component. **0b0000** RAS System Architecture v1.0. **0b0001** RAS System Architecture v1.1. As 0b0000 and also: • Simplifies ext-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.	xxxx
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **0b0000** RAS System Architecture v1. All other values are reserved. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12]. This field reads as 0b0000.	xxxx
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **0b101000000000** RAS System Architecture. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHPART is ARCHID[11:0]. This field reads as 0xA00.	12{x}

Accessibility

Component	Offset	Instance	Range
Core RAS	0xFBC	ERRDEVARCH	None

This interface is accessible as follows:

RO

B.3.14 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

Core RAS

Register offset

0xFC8

Access type

RO

Reset value

xxxx xxxx xxxx xxxx 0000 0000 0000 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-39: core_ras.errdevid bit assignments

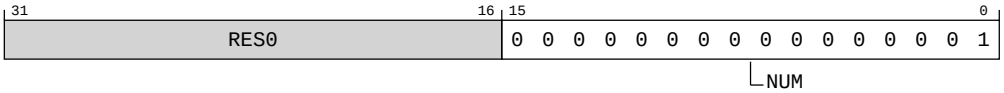


Table B-80: ERRDEVID bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[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. **0b0000000000000001** One record present.	0x0001

Accessibility

Component	Offset	Instance	Range
Core RAS	0xFC8	ERRDEVID	None

This interface is accessible as follows:

RO

B.3.15 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

Core RAS

Register offset

0xFD0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-40: core_ras.errpidr4 bit assignments

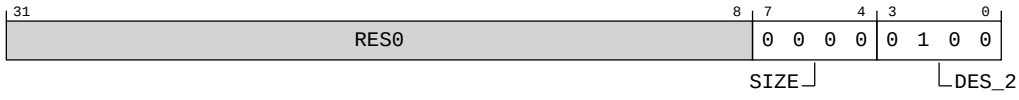


Table B-82: ERRPIDR4 bit descriptions

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
Core RAS	0xFD0	ERRPIDR4	None

This interface is accessible as follows:

RO

B.3.16 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

Core RAS

Register offset

0xFE0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1000 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-41: core_ras.errpidr0 bit assignments

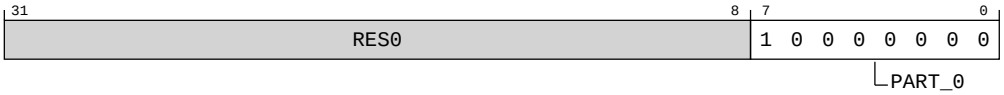


Table B-84: ERRPIDR0 bit descriptions

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, it is stored in ext-ERRPIDR1.PART_1 and ERRPIDR0.PART_0. There are 8 bits, ext-ERRPIDR2.REVISION and ext-ERRPIDR3.REVAND, available to define the revision of the component. • If a 16-bit part number is used, it is stored in ext-ERRPIDR2.PART_2, ext-ERRPIDR1.PART_1 and ERRPIDR0.PART_0. There are 4 bits, ext-ERRPIDR3.REVISION, available to define the revision of the component. **0b10000000** Cortex-A520	0x80

Accessibility

Component	Offset	Instance	Range
Core RAS	0xFE0	ERRPIDR0	None

This interface is accessible as follows:

RO

B.3.17 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

Core RAS

Register offset

0xFE4

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-42: core_ras.errpidr1 bit assignments

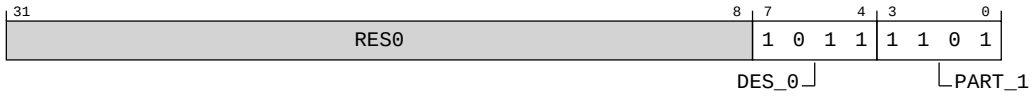


Table B-86: ERRPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7:4]	DES_0	Designer, JEP106 identification code, bits [3:0]. ERRPIDR1.DES_0 and ext-ERRPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. **0b1011** Arm Limited	0b1011
[3:0]	PART_1	Part number, bits [11:8]. The part number is selected by the designer of the component. The designer chooses whether to use a 12-bit or a 16-bit part number: • If a 12-bit part number is used, it is stored in ERRPIDR1.PART_1 and ext-ERRPIDR0.PART_0. There are 8 bits, ext-ERRPIDR2.REVISION and ext-ERRPIDR3.REVAND, available to define the revision of the component. • If a 16-bit part number is used, it is stored in ext-ERRPIDR2.PART_2, ERRPIDR1.PART_1 and ext-ERRPIDR0.PART_0. There are 4 bits, ext-ERRPIDR3.REVISION, available to define the revision of the component. **0b1101** Cortex-A520	0b1101

Accessibility

Component	Offset	Instance	Range
Core RAS	0xFE4	ERRPIDR1	None

This interface is accessible as follows:

RO

B.3.18 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

Core RAS

Register offset

0xFE8

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-43: core_ras.errpidr2 bit assignments

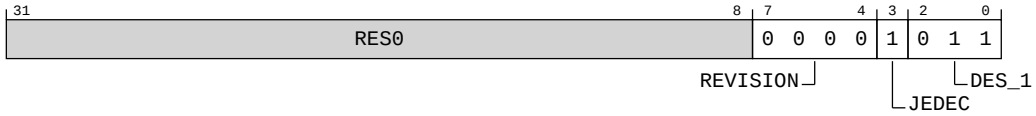


Table B-88: ERRPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Component major revision. ERRPIDR2.REVISION and ext-ERRPIDR3.REVAND together form the revision number of the component, with ERRPIDR2.REVISION being the most significant part and ext-ERRPIDR3.REVAND the least significant part. When a component is changed, ERRPIDR2.REVISION or ext-ERRPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. ext-ERRPIDR3.REVAND should be set to `0b0000` when ERRPIDR2.REVISION is increased. **`0b0000`** r0p2	`0b0000`
[3]	JEDEC	JEDEC-assigned JEP106 implementer code is used. **`0b1`**	`0b1`
[2:0]	DES_1	Designer, JEP106 identification code, bits [6:4]. ext-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
Core RAS	0xFE8	ERRPIDR2	None

This interface is accessible as follows:

RO

B.3.19 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

Core RAS

Register offset

0xFEC

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0010 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-44: core_ras.errpidr3 bit assignments

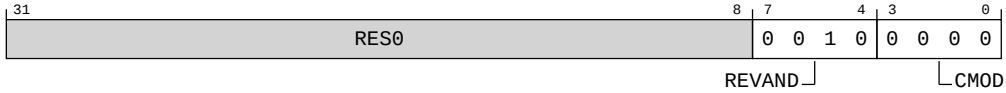


Table B-90: ERRPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	REVAND	Component minor revision. ext-ERRPIDR2.REVISION and ERRPIDR3.REVAND together form the revision number of the component, with ext-ERRPIDR2.REVISION being the most significant part and ERRPIDR3.REVAND the least significant part. When a component is changed, ext-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 ext-ERRPIDR2.REVISION is increased. **`0b0010`** r0p2	`0b0010`
[3:0]	CMOD	Customer Modified. Indicates the component has been modified. A value of `0b0000` means the component is not modified from the original design. Any other value means the component has been modified in an **IMPLEMENTATION DEFINED** way. **`0b0000`**	`0b0000`

Accessibility

Component	Offset	Instance	Range
Core RAS	0xFEC	ERRPIDR3	None

This interface is accessible as follows:

RO

B.3.20 ERRCIDR0, Component 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

ERRCIDR0 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

Core RAS

Register offset

0xFF0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-45: core_ras.errcidr0 bit assignments

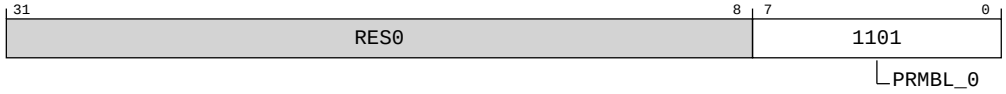


Table B-92: ERRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Component identification preamble, segment 0. **0b00001101**	0x0D

Accessibility

Component	Offset	Instance	Range
Core RAS	0xFF0	ERRCIDR0	None

This interface is accessible as follows:

RO

B.3.21 ERRCIDR1, Component 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

ERRCIDR1 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

Core RAS

Register offset

0xFF4

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-46: core_ras.errcidr1 bit assignments

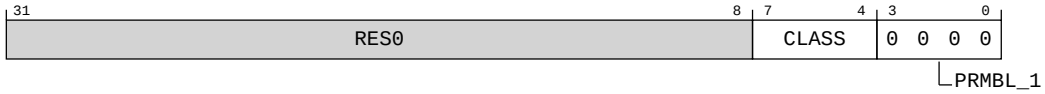


Table B-94: ERRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1111** Generic peripheral with **IMPLEMENTATION DEFINED** register layout. Other values are defined by the CoreSight Architecture. This field reads as 0xF.	xxxx
[3:0]	PRMBL_1	Component identification preamble, segment 1. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
Core RAS	0xFF4	ERRCIDR1	None

This interface is accessible as follows:

RO

B.3.22 ERRCIDR2, Component 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

ERRCIDR2 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

Core RAS

Register offset

0xFF8

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-47: core_ras.errcidr2 bit assignments

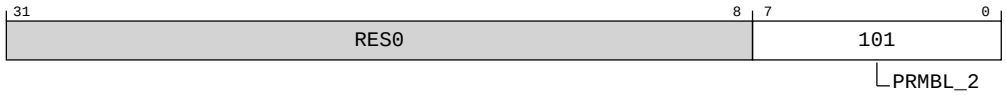


Table B-96: ERRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Component identification preamble, segment 2. **0b00000101**	0x05

Accessibility

Component	Offset	Instance	Range
Core RAS	0xFF8	ERRCIDR2	None

This interface is accessible as follows:

RO

B.3.23 ERRCIDR3, Component 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

ERRCIDR3 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

Core RAS

Register offset

0xFFC

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-48: core_ras.errcidr3 bit assignments

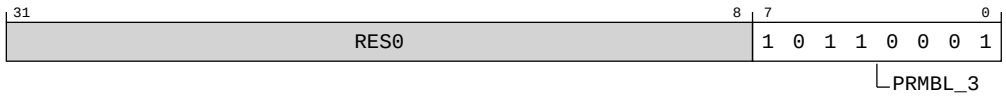


Table B-98: ERRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Component identification preamble, segment 3. **0b10110001**	0xB1

Accessibility

Component	Offset	Instance	Range
Core RAS	0xFFC	ERRCIDR3	None

This interface is accessible as follows:

RO

B.4 External PMU registers summary

The 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-100: PMU registers summary

Offset	Name	Reset	Width	Description
0x0	PMEVCNTR0_EL0	—	64-bit	Performance Monitors Event Count Registers
0x8	PMEVCNTR1_EL0	—	64-bit	Performance Monitors Event Count Registers
0x10	PMEVCNTR2_EL0	—	64-bit	Performance Monitors Event Count Registers
0x18	PMEVCNTR3_EL0	—	64-bit	Performance Monitors Event Count Registers
0x20	PMEVCNTR4_EL0	—	64-bit	Performance Monitors Event Count Registers
0x28	PMEVCNTR5_EL0	—	64-bit	Performance Monitors Event Count Registers
0x30	PMEVCNTR6_EL0	—	64-bit	Performance Monitors Event Count Registers
0x38	PMEVCNTR7_EL0	—	64-bit	Performance Monitors Event Count Registers
0x40	PMEVCNTR8_EL0	—	64-bit	Performance Monitors Event Count Registers
0x48	PMEVCNTR9_EL0	—	64-bit	Performance Monitors Event Count Registers
0x50	PMEVCNTR10_EL0	—	64-bit	Performance Monitors Event Count Registers
0x58	PMEVCNTR11_EL0	—	64-bit	Performance Monitors Event Count Registers
0x60	PMEVCNTR12_EL0	—	64-bit	Performance Monitors Event Count Registers

Offset	Name	Reset	Width	Description
0x68	PMEVCNTR13_EL0	—	64-bit	Performance Monitors Event Count Registers
0x70	PMEVCNTR14_EL0	—	64-bit	Performance Monitors Event Count Registers
0x78	PMEVCNTR15_EL0	—	64-bit	Performance Monitors Event Count Registers
0x80	PMEVCNTR16_EL0	—	64-bit	Performance Monitors Event Count Registers
0x88	PMEVCNTR17_EL0	—	64-bit	Performance Monitors Event Count Registers
0x90	PMEVCNTR18_EL0	—	64-bit	Performance Monitors Event Count Registers
0x98	PMEVCNTR19_EL0	—	64-bit	Performance Monitors Event Count Registers
0x0F8	PMCCNTR_EL0[31:0]	—	32-bit	Performance Monitors Cycle Counter
0x0FC	PMCCNTR_EL0[63:32]	—	32-bit	Performance Monitors Cycle Counter
0x200	PMPCSR[31:0]	—	32-bit	Program Counter Sample Register
0x204	PMPCSR[63:32]	—	32-bit	Program Counter Sample Register
0x220	PMPCSR[31:0]	—	32-bit	Program Counter Sample Register
0x224	PMPCSR[63:32]	—	32-bit	Program Counter Sample Register
0x208	PMCID1SR	—	32-bit	CONTEXTIDR_EL1 Sample Register
0x228	PMCID1SR	—	32-bit	CONTEXTIDR_EL1 Sample Register
0x20C	PMVIDSR	—	32-bit	VMID Sample Register
0x22C	PMCID2SR	—	32-bit	CONTEXTIDR_EL2 Sample Register
0x400	PMEVTYPER0_EL0	—	32-bit	Performance Monitors Event Type Registers
0x404	PMEVTYPER1_EL0	—	32-bit	Performance Monitors Event Type Registers
0x408	PMEVTYPER2_EL0	—	32-bit	Performance Monitors Event Type Registers
0x40C	PMEVTYPER3_EL0	—	32-bit	Performance Monitors Event Type Registers
0x410	PMEVTYPER4_EL0	—	32-bit	Performance Monitors Event Type Registers
0x414	PMEVTYPER5_EL0	—	32-bit	Performance Monitors Event Type Registers
0x418	PMEVTYPER6_EL0	—	32-bit	Performance Monitors Event Type Registers
0x41C	PMEVTYPER7_EL0	—	32-bit	Performance Monitors Event Type Registers
0x420	PMEVTYPER8_EL0	—	32-bit	Performance Monitors Event Type Registers
0x424	PMEVTYPER9_EL0	—	32-bit	Performance Monitors Event Type Registers
0x428	PMEVTYPER10_EL0	—	32-bit	Performance Monitors Event Type Registers
0x42C	PMEVTYPER11_EL0	—	32-bit	Performance Monitors Event Type Registers
0x430	PMEVTYPER12_EL0	—	32-bit	Performance Monitors Event Type Registers
0x434	PMEVTYPER13_EL0	—	32-bit	Performance Monitors Event Type Registers
0x438	PMEVTYPER14_EL0	—	32-bit	Performance Monitors Event Type Registers
0x43C	PMEVTYPER15_EL0	—	32-bit	Performance Monitors Event Type Registers
0x440	PMEVTYPER16_EL0	—	32-bit	Performance Monitors Event Type Registers
0x444	PMEVTYPER17_EL0	—	32-bit	Performance Monitors Event Type Registers
0x448	PMEVTYPER18_EL0	—	32-bit	Performance Monitors Event Type Registers
0x44C	PMEVTYPER19_EL0	—	32-bit	Performance Monitors Event Type Registers
0x47C	PMCCFILTR_EL0	—	32-bit	Performance Monitors Cycle Counter Filter Register
0x600	PMPCSSR	—	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	—	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register

Offset	Name	Reset	Width	Description
0x610	PMSSSR	—	32-bit	PMU Snapshot Status Register
0x614	PMOVSSR	—	32-bit	PMU Overflow Status Snapshot Register
0x618	PMCCNTSR	—	64-bit	PMU Cycle Counter Snapshot Register
0x620	PMEVCNTSR0	—	64-bit	PMU Event Counter Snapshot Register
0x628	PMEVCNTSR1	—	64-bit	PMU Event Counter Snapshot Register
0x630	PMEVCNTSR2	—	64-bit	PMU Event Counter Snapshot Register
0x638	PMEVCNTSR3	—	64-bit	PMU Event Counter Snapshot Register
0x640	PMEVCNTSR4	—	64-bit	PMU Event Counter Snapshot Register
0x648	PMEVCNTSR5	—	64-bit	PMU Event Counter Snapshot Register
0x650	PMEVCNTSR6	—	64-bit	PMU Event Counter Snapshot Register
0x658	PMEVCNTSR7	—	64-bit	PMU Event Counter Snapshot Register
0x660	PMEVCNTSR8	—	64-bit	PMU Event Counter Snapshot Register
0x668	PMEVCNTSR9	—	64-bit	PMU Event Counter Snapshot Register
0x670	PMEVCNTSR10	—	64-bit	PMU Event Counter Snapshot Register
0x678	PMEVCNTSR11	—	64-bit	PMU Event Counter Snapshot Register
0x680	PMEVCNTSR12	—	64-bit	PMU Event Counter Snapshot Register
0x688	PMEVCNTSR13	—	64-bit	PMU Event Counter Snapshot Register
0x690	PMEVCNTSR14	—	64-bit	PMU Event Counter Snapshot Register
0x698	PMEVCNTSR15	—	64-bit	PMU Event Counter Snapshot Register
0x6A0	PMEVCNTSR16	—	64-bit	PMU Event Counter Snapshot Register
0x6A8	PMEVCNTSR17	—	64-bit	PMU Event Counter Snapshot Register
0x6B0	PMEVCNTSR18	—	64-bit	PMU Event Counter Snapshot Register
0x6B8	PMEVCNTSR19	—	64-bit	PMU Event Counter Snapshot Register
0xC00	PMCNTENSET_EL0	—	32-bit	Performance Monitors Count Enable Set register
0xC20	PMCNTENCLR_EL0	—	32-bit	Performance Monitors Count Enable Clear register
0xC40	PMINTENSET_EL1	—	32-bit	Performance Monitors Interrupt Enable Set register
0xC60	PMINTENCLR_EL1	—	32-bit	Performance Monitors Interrupt Enable Clear register
0xC80	PMOVSCLR_EL0	—	32-bit	Performance Monitors Overflow Flag Status Clear register
0xCC0	PMOVSSET_EL0	—	32-bit	Performance Monitors Overflow Flag Status Set register
0xE00	PMCFGR	—	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_EL0	—	32-bit	Performance Monitors Control Register
0xE20	PMCEID0	—	32-bit	Performance Monitors Common Event Identification register 0
0xE24	PMCEID1	—	32-bit	Performance Monitors Common Event Identification register 1
0xE28	PMCEID2	—	32-bit	Performance Monitors Common Event Identification register 2
0xE2C	PMCEID3	—	32-bit	Performance Monitors Common Event Identification register 3
0xE30	PMSSCR	—	32-bit	PMU Snapshot Capture Register
0xE40	PMMIR	—	32-bit	Performance Monitors Machine Identification Register
0xFA8	PMDEVAFF0	—	32-bit	Performance Monitors Device Affinity register 0
0xFAC	PMDEVAFF1	—	32-bit	Performance Monitors Device Affinity register 1
0xFB0	PMLAR	—	32-bit	Performance Monitors Lock Access Register

Offset	Name	Reset	Width	Description
0xFB4	PMLSR	–	32-bit	Performance Monitors Lock Status Register
0xFB8	PMAUTHSTATUS	–	32-bit	Performance Monitors Authentication Status register
0xFBC	PMDEVARCH	–	32-bit	Performance Monitors Device Architecture register
0xFC8	PMDEVID	–	32-bit	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	–	32-bit	Performance Monitors Device Type register
0xFD0	PMPIDR4	–	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	–	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	–	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	–	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	–	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	–	32-bit	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	–	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	–	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	–	32-bit	Performance Monitors Component Identification Register 3

B.4.1 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

Read

R

Write

RESERVED

Reset value

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

Figure B-49: ext_pmpcssr bit assignments

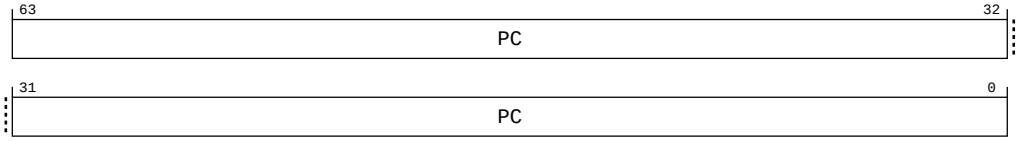


Table B-101: PMPCSSR bit descriptions

Bits	Name	Description	Reset
[63:0]	PC	Sampled PC. The instruction address for the sampled instruction. The sampled instruction must be an instruction recently executed by the PE. The architecture does not require that all instructions are eligible for sampling. However, it must be possible to reference instructions at branch targets. The branch target for a conditional branch instruction that fails its Condition code check is the instruction following the conditional branch target. The sampled instruction must be architecturally executed. However, in exceptional circumstances, such as a change in security state or other boundary condition, it is permissible to sample an instruction that was speculatively executed and not architecturally executed. **Note:** The ARM architecture does not define recently executed.	`64{x}`

Accessibility

PMPCSSR is set to an UNKNOWN value by a read of the EDPCSRlo or PMPCSR[31:0] register.

Component	Offset
PMU	0x600

This interface is accessible as follows:

RO

B.4.2 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

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-50: ext_pmcidssr bit assignments

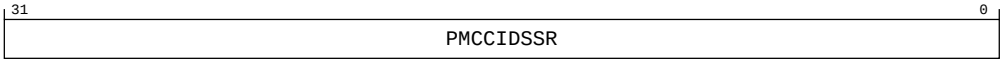


Table B-103: PMCIDSSR bit descriptions

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
PMU	0x608

This interface is accessible as follows:

RO

B.4.3 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

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx1

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-51: ext_pmsssr bit assignments

Table B-105: PMSSSR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	NC	No capture. Indicates whether the PMU counters have been captured. **0b0** PMU counters captured. **0b1** PMU counters not captured. The event counters are only not captured by the PE in the event of a security violation. The external Monitor is responsible for keeping track of whether it managed to capture the snapshot registers from the PE. PMSSR.NC does not reflect the status of the captured Program Counter Sample registers. PMSSR.NC is reset to 1 by PE Warm reset, but is overwritten at the first capture. Tools need to be aware that capturing over reset or power-down might lose data, as they are reliant on software saving and restoring the PMU state (including PMSSCR). There is no sampled sticky reset bit.	0b1

Accessibility

Component	Offset
PMU	0x610

This interface is accessible as follows:

RO

B.4.4 PMOVSSR, PMU Overflow Status Snapshot Register

Captured copy of PMOVSR. Once captured, the value in PMOVSSR is unaffected by writes to PMOVSSET_EL0 and PMOVSCLR_EL0.

Configurations

If PMSSRR is not implemented, PMOVSSR is optional.

Attributes

Width

32

Component

PMU

Register offset

0x614

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-52: ext_pmovssr bit assignments

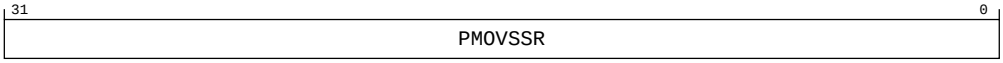


Table B-107: PMOVSSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMOVSSR	PMOVSR sample. Sampled overflow status.	32{x}

Accessibility

Component	Offset
PMU	0x614

This interface is accessible as follows:

RO

B.4.5 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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-53: ext_pmccntsr bit assignments

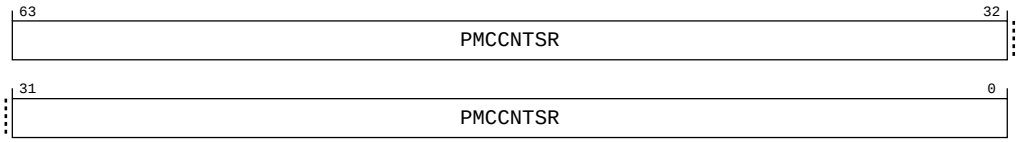


Table B-109: PMCCNTSR bit descriptions

Bits	Name	Description	Reset
[63:0]	PMCCNTSR	PMCCNTR_EL0 sample. Sampled cycle count.	64{x}

Accessibility

Component	Offset
PMU	0x618

This interface is accessible as follows:

RO

B.4.6 PMEVCNTSR0, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x620

Access type

RO

Reset value

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

Figure B-54: ext_pmevcntsr0 bit assignments

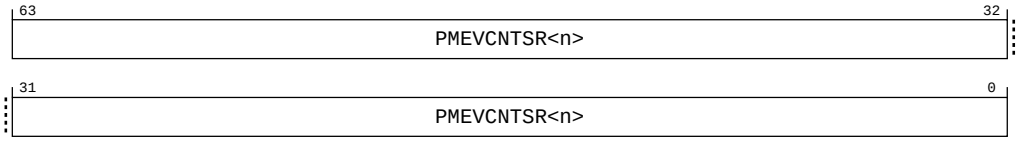


Table B-111: PMEVCNTSR0 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x620	PMEVCNTSR0	None

This interface is accessible as follows:

RO

B.4.7 PMEVCNTSR1, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x628

Access type

RO

Reset value

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

Figure B-55: ext_pmevcntsr1 bit assignments

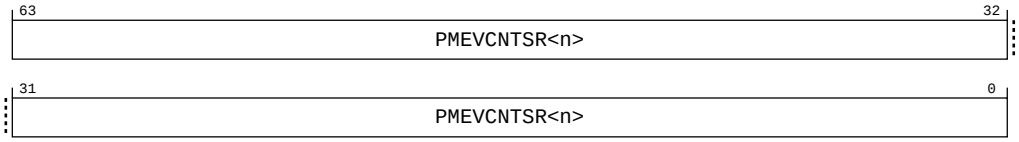


Table B-113: PMEVCNTSR1 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x628	PMEVCNTSR1	None

This interface is accessible as follows:

RO

B.4.8 PMEVCNTSR2, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x630

Access type

RO

Reset value

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

Figure B-56: ext_pmevcntsr2 bit assignments

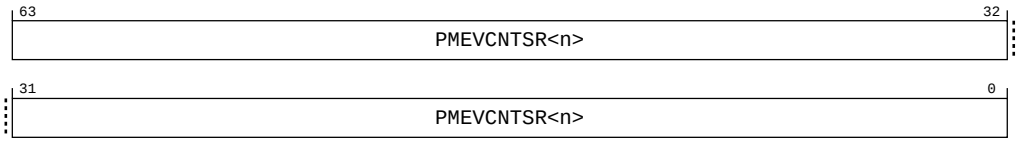


Table B-115: PMEVCNTSR2 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x630	PMEVCNTSR2	None

This interface is accessible as follows:

RO

B.4.9 PMEVCNTSR3, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x638

Access type

RO

Reset value

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

Figure B-57: ext_pmevcntsr3 bit assignments

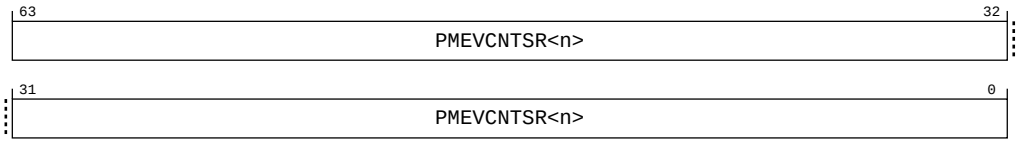


Table B-117: PMEVCNTSR3 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x638	PMEVCNTSR3	None

This interface is accessible as follows:

RO

B.4.10 PMEVCNTSR4, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x640

Access type

RO

Reset value

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

Figure B-58: ext_pmevcntsr4 bit assignments

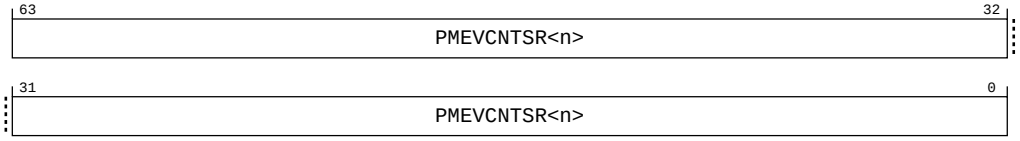


Table B-119: PMEVCNTSR4 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	`64{x}`

Accessibility

Component	Offset	Instance	Range
PMU	0x640	PMEVCNTSR4	None

This interface is accessible as follows:

RO

B.4.11 PMEVCNTSR5, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x648

Access type

RO

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-59: ext_pmevcntsr5 bit assignments

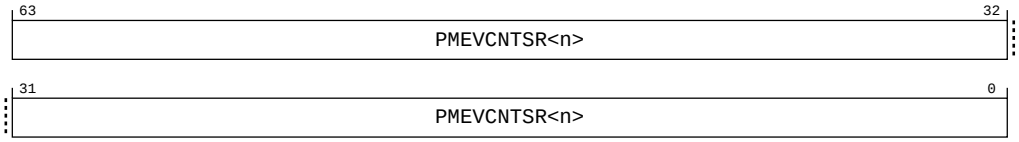


Table B-121: PMEVCNTSR5 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x648	PMEVCNTSR5	None

This interface is accessible as follows:

RO

B.4.12 PMEVCNTSR6, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x650

Access type

RO

Reset value

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

Figure B-60: ext_pmevcntsr6 bit assignments

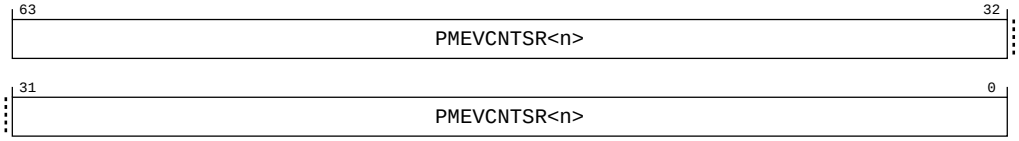


Table B-123: PMEVCNTSR6 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x650	PMEVCNTSR6	None

This interface is accessible as follows:

RO

B.4.13 PMEVCNTSR7, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x658

Access type

RO

Reset value

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

Figure B-61: ext_pmevcntsr7 bit assignments

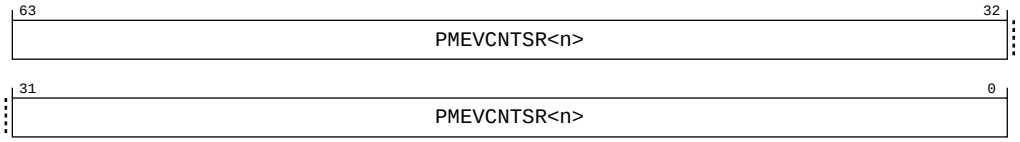


Table B-125: PMEVCNTSR7 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x658	PMEVCNTSR7	None

This interface is accessible as follows:

RO

B.4.14 PMEVCNTSR8, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x660

Access type

RO

Reset value

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

Figure B-62: ext_pmevcntsr8 bit assignments

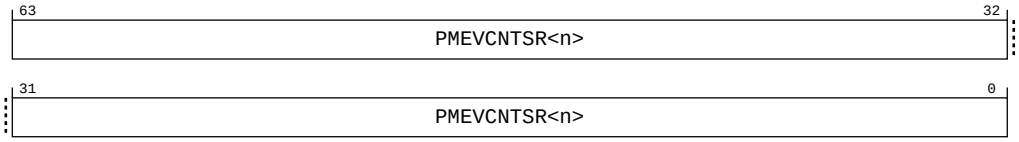


Table B-127: PMEVCNTSR8 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x660	PMEVCNTSR8	None

This interface is accessible as follows:

RO

B.4.15 PMEVCNTSR9, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x668

Access type

RO

Reset value

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

Figure B-63: ext_pmevcntsr9 bit assignments

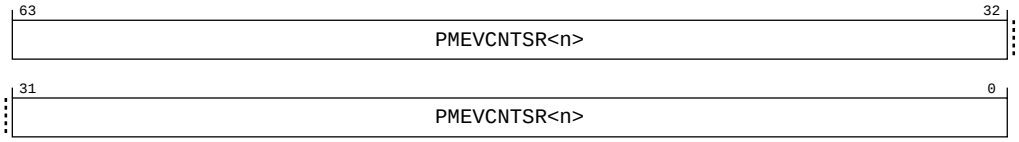


Table B-129: PMEVCNTSR9 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x668	PMEVCNTSR9	None

This interface is accessible as follows:

RO

B.4.16 PMEVCNTSR10, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x670

Access type

RO

Reset value

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

Figure B-64: ext_pmevcntsr10 bit assignments

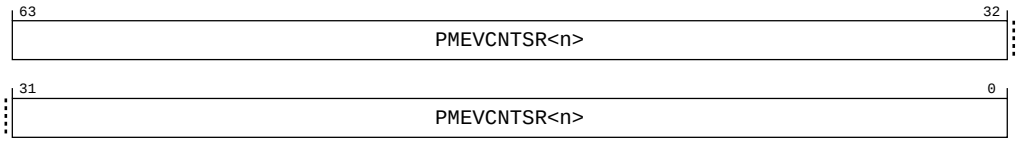


Table B-131: PMEVCNTSR10 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x670	PMEVCNTSR10	None

This interface is accessible as follows:

RO

B.4.17 PMEVCNTSR11, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x678

Access type

RO

Reset value

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

Figure B-65: ext_pmevcntsr11 bit assignments

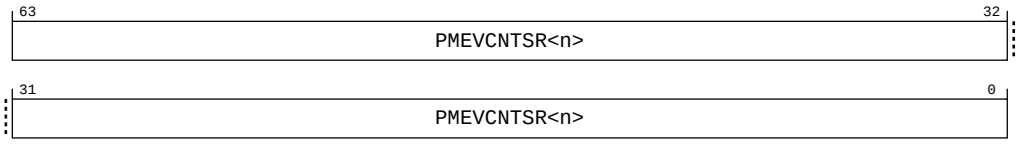


Table B-133: PMEVCNTSR11 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x678	PMEVCNTSR11	None

This interface is accessible as follows:

RO

B.4.18 PMEVCNTSR12, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x680

Access type

RO

Reset value

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

Figure B-66: ext_pmevcntsr12 bit assignments

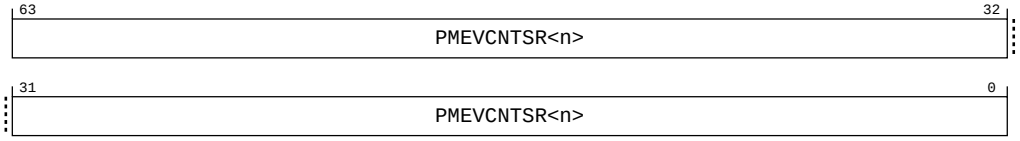


Table B-135: PMEVCNTSR12 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x680	PMEVCNTSR12	None

This interface is accessible as follows:

RO

B.4.19 PMEVCNTSR13, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x688

Access type

RO

Reset value

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

Figure B-67: ext_pmevcntsr13 bit assignments

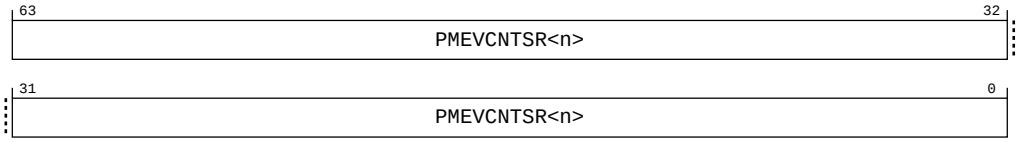


Table B-137: PMEVCNTSR13 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x688	PMEVCNTSR13	None

This interface is accessible as follows:

RO

B.4.20 PMEVCNTSR14, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x690

Access type

RO

Reset value

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

Figure B-68: ext_pmevcntsr14 bit assignments

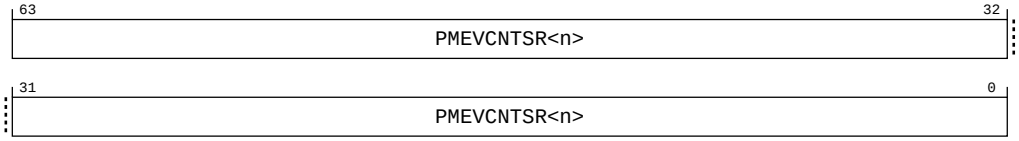


Table B-139: PMEVCNTSR14 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x690	PMEVCNTSR14	None

This interface is accessible as follows:

RO

B.4.21 PMEVCNTSR15, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x698

Access type

RO

Reset value

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

Figure B-69: ext_pmevcntsr15 bit assignments

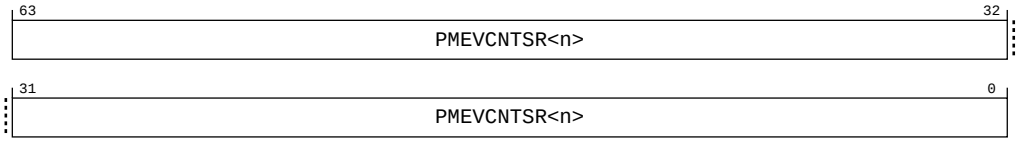


Table B-141: PMEVCNTSR15 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x698	PMEVCNTSR15	None

This interface is accessible as follows:

RO

B.4.22 PMEVCNTSR16, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6A0

Access type

RO

Reset value

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

Figure B-70: ext_pmevcntsr16 bit assignments

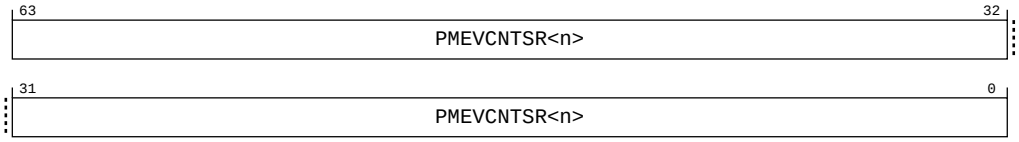


Table B-143: PMEVCNTSR16 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6A0	PMEVCNTSR16	None

This interface is accessible as follows:

RO

B.4.23 PMEVCNTSR17, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6A8

Access type

RO

Reset value

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

Figure B-71: ext_pmevcntsr17 bit assignments

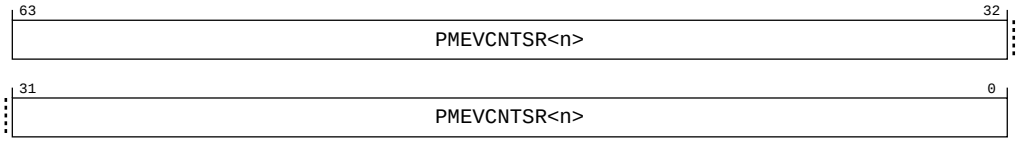


Table B-145: PMEVCNTSR17 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6A8	PMEVCNTSR17	None

This interface is accessible as follows:

RO

B.4.24 PMEVCNTSR18, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6B0

Access type

RO

Reset value

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

Figure B-72: ext_pmevcntsr18 bit assignments

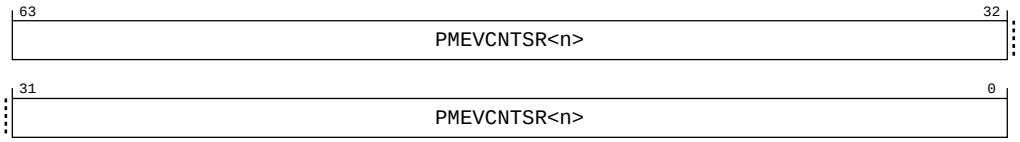


Table B-147: PMEVCNTSR18 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6B0	PMEVCNTSR18	None

This interface is accessible as follows:

RO

B.4.25 PMEVCNTSR19, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6B8

Access type

RO

Reset value

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

Figure B-73: ext_pmevcntsr19 bit assignments

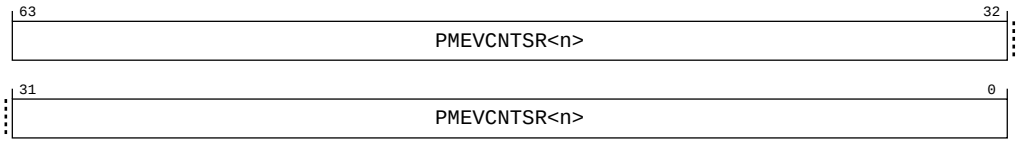


Table B-149: PMEVCNTSR19 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6B8	PMEVCNTSR19	None

This interface is accessible as follows:

RO

B.4.26 PMCFGR, Performance Monitors Configuration Register

Contains PMU-specific configuration data.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE00

Access type

See bit descriptions

Reset value

0000 xxxx xxxx 0000 x111 1111 xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-74: ext_pmcfgr bit assignments

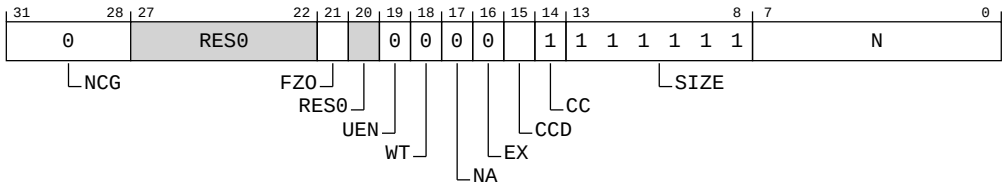


Table B-151: PMCFGR bit descriptions

Bits	Name	Description	Reset
[31:28]	NCG	This feature is not supported, so this field is **RAZ**. **0b0000**	0b0000
[27:22]	**RES0**	Reserved	**RES0**
[21]	FZO	Freeze-on-overflow supported. Defined values are: **0b1** Freeze-on-overflow mechanism is supported. ext-PMCR_EL0.FZO is RW.	x
[20]	**RES0**	Reserved	**RES0**
[19]	UEN	User-mode Enable Register supported. AArch64-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. Value is **IMPLEMENTATION DEFINED**. **0b0** ext-PMCR_EL0.X is **RES0**.	0b0
[15]	CCD	Cycle counter has prescale. This field is **RAZ** **0b0** ext-PMCR_EL0.D is **RES0**.	x
[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, the largest counter is 64-bits, so the value of this field is 0b111111. This field is used by software to determine the spacing of the counters in the memory-map. From Armv8, the counters are a doubleword-aligned addresses. **0b111111**	0b111111
[7:0]	N	Number of counters implemented in addition to the cycle counter, ext-PMCCNTR_EL0. The maximum number of event counters is 31. **0b00010100** Twenty PMU Counters Implemented **0b00000110** Six PMU Counters Implemented	8{x}

Access

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE00	PMCFGR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.4.27 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

External register PMCR_EL0 bits [7:0] are architecturally mapped to AArch64 System register A.7.2 PMCR_EL0, Performance Monitors Control Register on page 361 bits [7:0].

Attributes

Width

32

Component

PMU

Register offset

0xE04

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0xxx xxx0 x000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-75: ext_pmcr_el0 bit assignments

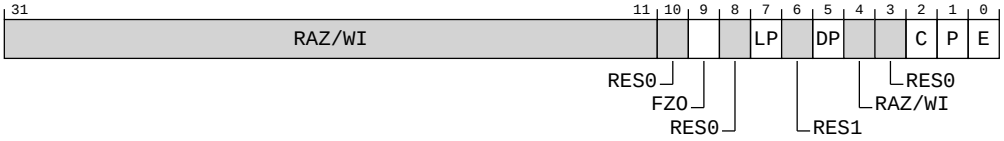


Table B-153: PMCR_EL0 bit descriptions

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. In the description of this field: • If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, PMN is PMCR_EL0.N. **0b0** Do not freeze on overflow. **0b1** Event counter ext-PMEVCNTR<n>_EL0 does not count when AArch64-PMOVSCLR_EL0[(PMN-1):0] is nonzero and n is in the range of affected event counters. If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)]. This field does not affect the operation of other event counters and AArch64-PMCCNTR_EL0. The operation of this field applies even when EL2 is disabled in the current Security state.	x
[8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7]	LP	Long event counter enable. Determines when unsigned overflow is recorded by an event counter overflow bit. In the description of this field: • If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, PMN is PMCR_EL0.N. **0b0** Event counter overflow on increment that causes unsigned overflow of ext-PMEVCNTR<n>_EL0[31:0]. **0b1** Event counter overflow on increment that causes unsigned overflow of ext-PMEVCNTR<n>_EL0[63:0]. If PMN is not 0, this bit affects the operation of event counters in the range [0 .. (PMN-1)]. The field does not affect the operation of other event counters and ext-PMCCNTR_EL0. The operation of this field applies even when EL2 is disabled in the current Security state.	x
[6]	**RES1**	Reserved	**RES1**
[5]	DP	Disable cycle counter when event counting is prohibited. The possible values of this bit are: **0b0** Cycle counting by ext-PMCCNTR_EL0 is not affected by this mechanism. **0b1** Cycle counting by ext-PMCCNTR_EL0 is disabled in prohibited regions: • If FEAT_PMUv3p1 is implemented, EL2 is implemented, and AArch64-MDCR_EL2.HPMD is 1, then cycle counting by ext-PMCCNTR_EL0 is disabled at EL2. • If FEAT_PMUv3p7 is implemented, EL3 is implemented and using AArch64, and AArch64-MDCR_EL3.MPMX is 1, then cycle counting by ext-PMCCNTR_EL0 is disabled at EL3. • If EL3 is implemented, AArch64-MDCR_EL3.SPME is 0, and either FEAT_PMUv3p7 is not implemented or AArch64-MDCR_EL3.MPMX is 0, then cycle counting by ext-PMCCNTR_EL0 is disabled at EL3 and in Secure state. If AArch64-MDCR_EL2.HPMN is not 0, this is when event counting by event counters in the range [0.. (AArch64-MDCR_EL2.HPMN-1)] is prohibited. For more information, see *Prohibiting event counting* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[4]	**RAZ/WI**	Reserved	**RAZ/WI**
[3]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[2]	C	Cycle counter reset. The effects of writing to this bit are: **0b0** No action. **0b1** Reset ext-PMCCNTR_EL0 to zero. **Note:** Resetting ext-PMCCNTR_EL0 does not change the cycle counter overflow bit. If FEAT_PMUv3p5 is implemented, the value of PMCR_EL0.LC is ignored, and bits [63:0] of the cycle counter are reset. Access to this field is: WO/**RAZ**	0b0
[1]	P	Event counter reset. The effects of writing to this bit are: **0b0** No action. **0b1** Reset all event counters, not including ext-PMCCNTR_EL0, to zero. **Note:** Resetting the event counters does not change the event counter overflow bits. If FEAT_PMUv3p5 is implemented, the value of AArch64-MDCR_EL2.HLP, or PMCR_EL0.LP is ignored and bits [63:0] of all affected event counters are reset. Access to this field is: WO/**RAZ**	0b0
[0]	E	Enable. In the description of this field: • If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, PMN is PMCR_EL0.N. **0b0** ext-PMCCNTR_EL0 is disabled and event counters ext-PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are disabled. **0b1** ext-PMCCNTR_EL0 and event counters ext-PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are enabled by ext-PMCNTENSET_EL0. If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)]. This field does not affect the operation of other event counters. The operation of this field applies even when EL2 is disabled in the current Security state.	0b0

Access

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.

Accessibility

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 IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess()

RW

Otherwise

ERROR

B.4.28 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

External register PMCEID0 bits [31:0] are architecturally mapped to AArch64 System register A.7.3 PMCEID0_EL0, Performance Monitors Common Event Identification register 0 on page 367 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xE20

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-76: ext_pmceid0 bit assignments

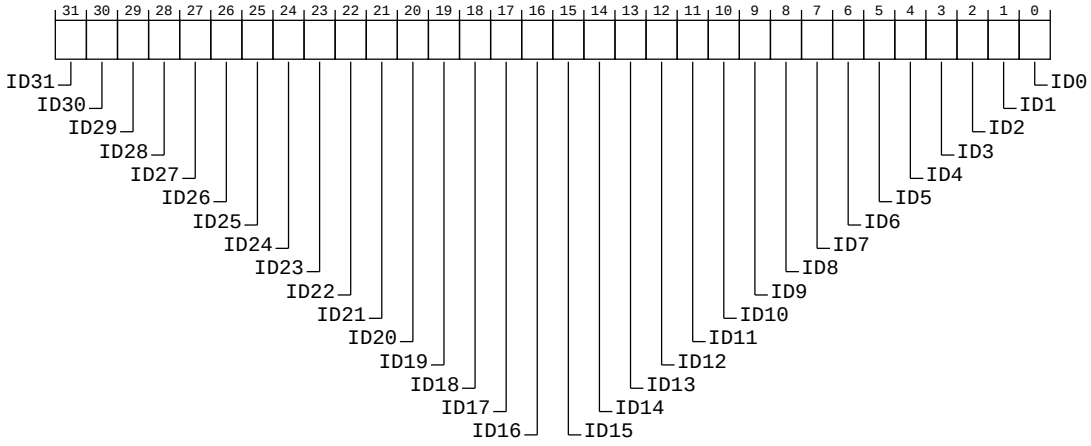


Table B-155: PMCEID0 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x1f) L1D_CACHE_ALLOCATE **0b0** The Common event is not implemented, or not counted.	x
[30]	ID30	ID30 corresponds to common event (0x1e) CHAIN **0b1** The Common event is implemented.	x
[29]	ID29	ID29 corresponds to common event (0x1d) BUS_CYCLES **0b1** The Common event is implemented.	x
[28]	ID28	ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED **0b1** The Common event is implemented.	x

Bits	Name	Description	Reset
[27]	ID27	ID27 corresponds to common event (0x1b) INST_SPEC **0b1** The Common event is implemented.	x
[26]	ID26	ID26 corresponds to common event (0x1a) MEMORY_ERROR **0b1** The Common event is implemented.	x
[25]	ID25	ID25 corresponds to common event (0x19) BUS_ACCESS **0b1** The Common event is implemented.	x
[24]	ID24	ID24 corresponds to common event (0x18) L2D_CACHE_WB **0b0** The common event is not implemented, or not counted. This value is reported if the Cortex-A520 complex is configured without an L2 cache. **0b1** The common event is implemented. This value is reported if the Cortex-A520 complex is configured with an L2 cache.	x
[23]	ID23	ID23 corresponds to common event (0x17) L2D_CACHE_REFILL **0b0** The common event is not implemented, or not counted. This value is reported if the Cortex-A520 complex is configured without an L2 cache. **0b1** The common event is implemented. This value is reported if the Cortex-A520 complex is configured with an L2 cache.	x
[22]	ID22	ID22 corresponds to common event (0x16) L2D_CACHE **0b0** The common event is not implemented, or not counted. This value is reported if both the Cortex-A520 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **0b1** The common event is implemented. This value is reported if either the Cortex-A520 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	x
[21]	ID21	ID21 corresponds to common event (0x15) L1D_CACHE_WB **0b1** The Common event is implemented.	x
[20]	ID20	ID20 corresponds to common event (0x14) L1I_CACHE **0b1** The Common event is implemented.	x
[19]	ID19	ID19 corresponds to common event (0x13) MEM_ACCESS **0b1** The Common event is implemented.	x
[18]	ID18	ID18 corresponds to common event (0x12) BR_PRED **0b1** The Common event is implemented.	x

Bits	Name	Description	Reset
[17]	ID17	ID17 corresponds to common event (0x11) CPU_CYCLES **0b1** The Common event is implemented.	x
[16]	ID16	ID16 corresponds to common event (0x10) BR_MIS_PRED **0b1** The Common event is implemented.	x
[15]	ID15	ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED **0b0** The Common event is not implemented, or not counted.	x
[14]	ID14	ID14 corresponds to common event (0xe) BR_RETURN_RETIRED **0b1** The Common event is implemented.	x
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED **0b1** The Common event is implemented.	x
[12]	ID12	ID12 corresponds to common event (0xc) PC_WRITE_RETIRED **0b1** The Common event is implemented.	x
[11]	ID11	ID11 corresponds to common event (0xb) CID_WRITE_RETIRED **0b1** The Common event is implemented.	x
[10]	ID10	ID10 corresponds to common event (0xa) EXC_RETURN **0b1** The Common event is implemented.	x
[9]	ID9	ID9 corresponds to common event (0x9) EXC_TAKEN **0b1** The Common event is implemented.	x
[8]	ID8	ID8 corresponds to common event (0x8) INST_RETIRED **0b1** The Common event is implemented.	x
[7]	ID7	ID7 corresponds to common event (0x7) ST_RETIRED **0b1** The Common event is implemented.	x
[6]	ID6	ID6 corresponds to common event (0x6) LD_RETIRED **0b1** The Common event is implemented.	x
[5]	ID5	ID5 corresponds to common event (0x5) L1D_TLB_REFILL **0b1** The Common event is implemented.	x
[4]	ID4	ID4 corresponds to common event (0x4) L1D_CACHE **0b1** The Common event is implemented.	x

Bits	Name	Description	Reset
[3]	ID3	ID3 corresponds to common event (0x3) L1D_CACHE_REFILL **0b1** The Common event is implemented.	x
[2]	ID2	ID2 corresponds to common event (0x2) L1I_TLB_REFILL **0b1** The Common event is implemented.	x
[1]	ID1	ID1 corresponds to common event (0x1) L1I_CACHE_REFILL **0b1** The Common event is implemented.	x
[0]	ID0	ID0 corresponds to common event (0x0) SW_INCR **0b1** The Common event is implemented.	x

Access

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

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 IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.4.29 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

External register PMCEID1 bits [31:0] are architecturally mapped to AArch64 System register A.7.4 PMCEID1_EL0, Performance Monitors Common Event Identification register 1 on page 374 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xE24

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-77: ext_pmceid1 bit assignments

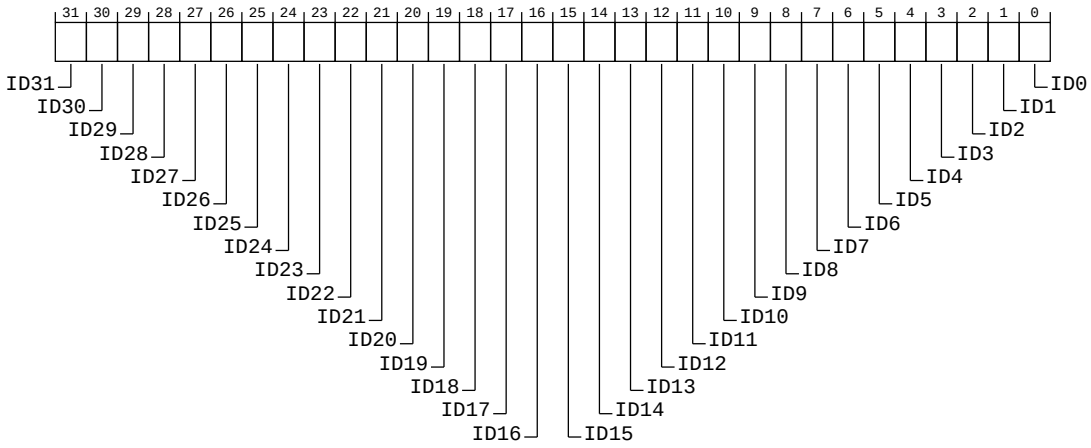


Table B-157: PMCEID1 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x3f) STALL_SLOT **0b1** The Common event is implemented.	x
[30]	ID30	ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND **0b1** The Common event is implemented.	x
[29]	ID29	ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND **0b1** The Common event is implemented.	x
[28]	ID28	ID28 corresponds to common event (0x3c) STALL **0b1** The Common event is implemented.	x
[27]	ID27	ID27 corresponds to common event (0x3b) OP_SPEC **0b1** The Common event is implemented.	x
[26]	ID26	ID26 corresponds to common event (0x3a) OP_RETIRED **0b1** The Common event is implemented.	x
[25]	ID25	ID25 corresponds to common event (`0x39`) L1D_CACHE_LMISS_RD **0b1** The Common event is implemented.	x
[24]	ID24	ID24 corresponds to common event (`0x38`) REMOTE_ACCESS_RD **0b1** The Common event is implemented.	x

Bits	Name	Description	Reset
[23]	ID23	ID23 corresponds to common event (0x37) LL_CACHE_MISS_RD **0b1** The Common event is implemented.	x
[22]	ID22	ID22 corresponds to common event (0x36) LL_CACHE_RD **0b1** The Common event is implemented.	x
[21]	ID21	ID21 corresponds to common event (0x35) ITLB_WLK **0b1** The Common event is implemented.	x
[20]	ID20	ID20 corresponds to common event (0x34) DTLB_WLK **0b1** The Common event is implemented.	x
[19]	ID19	ID19 corresponds to a Reserved Event event (0x33) **0b0** The Common event is not implemented, or not counted.	x
[18]	ID18	ID18 corresponds to a Reserved Event event (0x32) **0b0** The Common event is not implemented, or not counted.	x
[17]	ID17	ID17 corresponds to common event (0x31) REMOTE_ACCESS **0b0** The Common event is not implemented, or not counted.	x
[16]	ID16	ID16 corresponds to common event (0x30) L2I_TLB **0b0** The Common event is not implemented, or not counted.	x
[15]	ID15	ID15 corresponds to common event (0x2f) L2D_TLB **0b1** The Common event is implemented.	x
[14]	ID14	ID14 corresponds to common event (0x2e) L2I_TLB_REFILL **0b0** The Common event is not implemented, or not counted.	x
[13]	ID13	ID13 corresponds to common event (0x2d) L2D_TLB_REFILL **0b1** The Common event is implemented.	x
[12]	ID12	ID12 corresponds to common event (0x2c) Reserved **0b0** The Common event is not implemented, or not counted.	x

Bits	Name	Description	Reset
[11]	ID11	ID11 corresponds to common event (0x2b) L3D_CACHE **0b0** The common event is not implemented, or not counted. This value is reported if either the Cortex-A520 complex is configured without an L2 cache or the DSU is configured without an L3 cache. **0b1** The common event is implemented. This value is reported if both the Cortex-A520 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	x
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL **0b0** The Common event is not implemented, or not counted.	x
[9]	ID9	ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE **0b0** The Common event is not implemented, or not counted.	x
[8]	ID8	ID8 corresponds to common event (0x28) L2I_CACHE_REFILL **0b0** The Common event is not implemented, or not counted.	x
[7]	ID7	ID7 corresponds to common event (0x27) L2I_CACHE **0b0** The Common event is not implemented, or not counted.	x
[6]	ID6	ID6 corresponds to common event (0x26) L1I_TLB **0b1** The Common event is implemented.	x
[5]	ID5	ID5 corresponds to common event (0x25) L1D_TLB **0b1** The Common event is implemented.	x
[4]	ID4	ID4 corresponds to common event (0x24) STALL_BACKEND **0b1** The Common event is implemented.	x
[3]	ID3	ID3 corresponds to common event (0x23) STALL_FRONTEND **0b1** The Common event is implemented.	x
[2]	ID2	ID2 corresponds to common event (0x22) BR_MIS_PRED_RETIRED **0b1** The Common event is implemented.	x
[1]	ID1	ID1 corresponds to common event (0x21) BR_RETIRED **0b1** The Common event is implemented.	x

Bits	Name	Description	Reset
[0]	ID0	ID0 corresponds to common event (0x20) L2D_CACHE_ALLOCATE **0b0** The common event is not implemented, or not counted. This value is reported if the Cortex-A520 complex is configured without an L2 cache. **0b1** The common event is implemented. This value is reported if the Cortex-A520 complex is configured with an L2 cache.	x

Access

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

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 IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.4.30 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

External register PMCEID2 bits [31:0] are architecturally mapped to AArch64 System register A.7.3 PMCEID0_EL0, Performance Monitors Common Event Identification register 0 on page 367 bits [63:32].

Attributes

Width

32

Component

PMU

Register offset

0xE28

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-78: ext_pmceid2 bit assignments

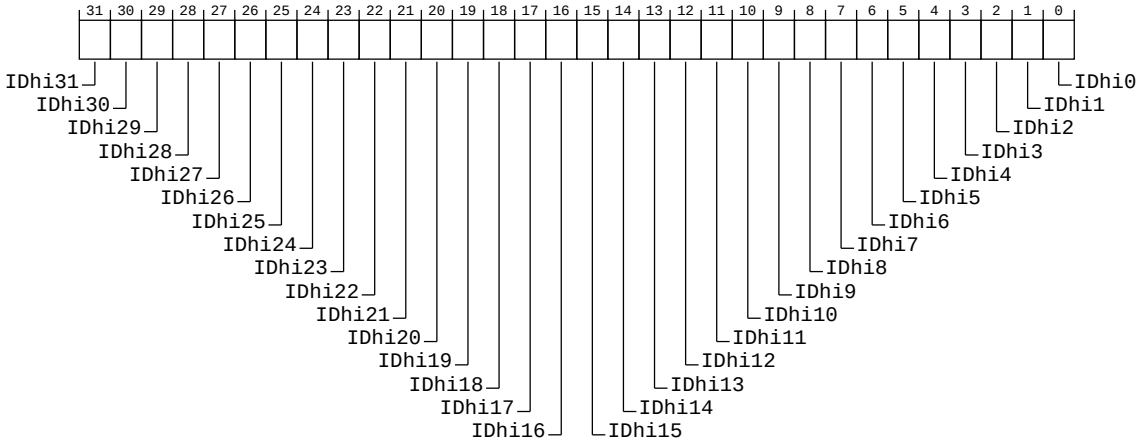


Table B-159: PMCEID2 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x401f) **0b0** The Common event is not implemented, or not counted.	x
[30]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x401e) **0b0** The Common event is not implemented, or not counted.	x

Bits	Name	Description	Reset
[29]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x401d) **0b0** The Common event is not implemented, or not counted.	x
[28]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x401c) **0b0** The Common event is not implemented, or not counted.	x
[27]	IDhi27	IDhi27 corresponds to common event (0x401b) CTI_TRIGOUT7 **0b1** The Common event is implemented.	x
[26]	IDhi26	IDhi26 corresponds to common event (0x401a) CTI_TRIGOUT6 **0b1** The Common event is implemented.	x
[25]	IDhi25	IDhi25 corresponds to common event (`0x4019`) CTI_TRIGOUT5 **0b1** The Common event is implemented.	x
[24]	IDhi24	IDhi24 corresponds to common event (`0x4018`) CTI_TRIGOUT4 **0b1** The Common event is implemented.	x
[23]	IDhi23	IDhi23 corresponds to a Reserved Event event (`0x4017`) **0b0** The Common event is not implemented, or not counted.	x
[22]	IDhi22	IDhi22 corresponds to a Reserved Event event (`0x4016`) **0b0** The Common event is not implemented, or not counted.	x
[21]	IDhi21	IDhi21 corresponds to a Reserved Event event (`0x4015`) **0b0** The Common event is not implemented, or not counted.	x
[20]	IDhi20	IDhi20 corresponds to a Reserved Event event (`0x4014`) **0b0** The Common event is not implemented, or not counted.	x
[19]	IDhi19	IDhi19 corresponds to common event (`0x4013`) TRCEXTOUT3 **0b1** The Common event is implemented.	x
[18]	IDhi18	IDhi18 corresponds to common event (`0x4012`) TRCEXTOUT2 **0b1** The Common event is implemented.	x
[17]	IDhi17	IDhi17 corresponds to common event (`0x4011`) TRCEXTOUT1 **0b1** The Common event is implemented.	x
[16]	IDhi16	IDhi16 corresponds to common event (`0x4010`) TRCEXTOUT0 **0b1** The Common event is implemented.	x

Bits	Name	Description	Reset
[15]	IDhi15	IDhi15 corresponds to common event (0x400f) PMU_HOVFS **0b0** The Common event is not implemented, or not counted.	x
[14]	IDhi14	IDhi14 corresponds to common event (0x400e) TRB_TRIG **0b1** The Common event is implemented.	x
[13]	IDhi13	IDhi13 corresponds to common event (0x400d) PMU_OVFS **0b0** The Common event is not implemented, or not counted.	x
[12]	IDhi12	IDhi12 corresponds to common event (0x400c) TRB_WRAP **0b1** The Common event is implemented.	x
[11]	IDhi11	IDhi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD **0b0** The common event is not implemented, or not counted. This value is reported if either the Cortex-A520 complex is configured without an L2 cache or the DSU is configured without an L3 cache. **0b1** The common event is implemented. This value is reported if both the Cortex-A520 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	x
[10]	IDhi10	IDhi10 corresponds to common event (0x400a) L2I_CACHE_LMISS **0b0** The Common event is not implemented, or not counted.	x
[9]	IDhi9	IDhi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD **0b0** The common event is not implemented, or not counted. This value is reported if both the Cortex-A520 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **0b1** The common event is implemented. This value is reported if either the Cortex-A520 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	x
[8]	IDhi8	IDhi8 corresponds to common event (0x4008) Reserved **0b0** The Common event is not implemented, or not counted.	x
[7]	IDhi7	IDhi7 corresponds to common event (0x4007) Reserved **0b0** The Common event is not implemented, or not counted.	x
[6]	IDhi6	IDhi6 corresponds to common event (0x4006) L1I_CACHE_LMISS **0b1** The Common event is implemented.	x
[5]	IDhi5	IDhi5 corresponds to common event (0x4005) STALL_BACKEND_MEM **0b1** The Common event is implemented.	x

Bits	Name	Description	Reset
[4]	IDhi4	IDhi4 corresponds to common event (0x4004) CNT_CYCLES **0b0** The Common event is not implemented, or not counted.	x
[3]	IDhi3	IDhi3 corresponds to common event (0x4003) SAMPLE_COLLISION **0b0** The Common event is not implemented, or not counted.	x
[2]	IDhi2	IDhi2 corresponds to common event (0x4002) SAMPLE_FILTRATE **0b0** The Common event is not implemented, or not counted.	x
[1]	IDhi1	IDhi1 corresponds to common event (0x4001) SAMPLE_FEED **0b0** The Common event is not implemented, or not counted.	x
[0]	IDhi0	IDhi0 corresponds to common event (0x4000) SAMPLE_POP **0b0** The Common event is not implemented, or not counted.	x

Access

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

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 IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.4.31 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

External register PMCEID3 bits [31:0] are architecturally mapped to AArch64 System register A.7.4 PMCEID1_EL0, Performance Monitors Common Event Identification register 1 on page 374 bits [63:32].

Attributes

Width

32

Component

PMU

Register offset

0xE2C

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-79: ext_pmceid3 bit assignments

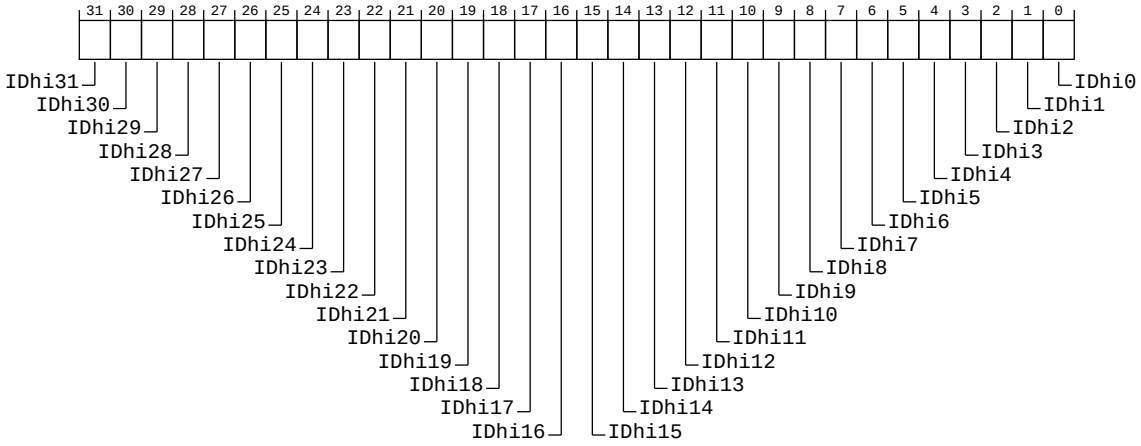


Table B-161: PMCEID3 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x403f) **0b0** The Common event is not implemented, or not counted.	x
[30]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x403e) **0b0** The Common event is not implemented, or not counted.	x
[29]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x403d) **0b0** The Common event is not implemented, or not counted.	x
[28]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x403c) **0b0** The Common event is not implemented, or not counted.	x
[27]	IDhi27	IDhi27 corresponds to a Reserved Event event (0x403b) **0b0** The Common event is not implemented, or not counted.	x
[26]	IDhi26	IDhi26 corresponds to a Reserved Event event (0x403a) **0b0** The Common event is not implemented, or not counted.	x
[25]	IDhi25	IDhi25 corresponds to a Reserved Event event (`0x4039`) **0b0** The Common event is not implemented, or not counted.	x
[24]	IDhi24	IDhi24 corresponds to a Reserved Event event (`0x4038`) **0b0** The Common event is not implemented, or not counted.	x

Bits	Name	Description	Reset
[23]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4037) **0b0** The Common event is not implemented, or not counted.	x
[22]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4036) **0b0** The Common event is not implemented, or not counted.	x
[21]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4035) **0b0** The Common event is not implemented, or not counted.	x
[20]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4034) **0b0** The Common event is not implemented, or not counted.	x
[19]	IDhi19	IDhi19 corresponds to a Reserved Event event (0x4033) **0b0** The Common event is not implemented, or not counted.	x
[18]	IDhi18	IDhi18 corresponds to a Reserved Event event (0x4032) **0b0** The Common event is not implemented, or not counted.	x
[17]	IDhi17	IDhi17 corresponds to a Reserved Event event (0x4031) **0b0** The Common event is not implemented, or not counted.	x
[16]	IDhi16	IDhi16 corresponds to a Reserved Event event (0x4030) **0b0** The Common event is not implemented, or not counted.	x
[15]	IDhi15	IDhi15 corresponds to a Reserved Event event (0x402f) **0b0** The Common event is not implemented, or not counted.	x
[14]	IDhi14	IDhi14 corresponds to a Reserved Event event (0x402e) **0b0** The Common event is not implemented, or not counted.	x
[13]	IDhi13	IDhi13 corresponds to a Reserved Event event (0x402d) **0b0** The Common event is not implemented, or not counted.	x
[12]	IDhi12	IDhi12 corresponds to a Reserved Event event (0x402c) **0b0** The Common event is not implemented, or not counted.	x
[11]	IDhi11	IDhi11 corresponds to a Reserved Event event (0x402b) **0b0** The Common event is not implemented, or not counted.	x
[10]	IDhi10	IDhi10 corresponds to a Reserved Event event (0x402a) **0b0** The Common event is not implemented, or not counted.	x

Bits	Name	Description	Reset
[9]	IDhi9	IDhi9 corresponds to a Reserved Event event (0x4029) **0b0** The Common event is not implemented, or not counted.	x
[8]	IDhi8	IDhi8 corresponds to a Reserved Event event (0x4028) **0b0** The Common event is not implemented, or not counted.	x
[7]	IDhi7	IDhi7 corresponds to a Reserved Event event (0x4027) **0b0** The Common event is not implemented, or not counted.	x
[6]	IDhi6	IDhi6 corresponds to common event (0x4026) MEM_ACCESS_WR_CHECKED **0b1** The Common event is implemented.	x
[5]	IDhi5	IDhi5 corresponds to common event (0x4025) MEM_ACCESS_RD_CHECKED **0b1** The Common event is implemented.	x
[4]	IDhi4	IDhi4 corresponds to common event (0x4024) MEM_ACCESS_CHECKED **0b1** The Common event is implemented.	x
[3]	IDhi3	IDhi3 corresponds to common event (0x4023) Reserved **0b0** The Common event is not implemented, or not counted.	x
[2]	IDhi2	IDhi2 corresponds to common event (0x4022) ST_ALIGN_LAT **0b1** The Common event is implemented.	x
[1]	IDhi1	IDhi1 corresponds to common event (0x4021) LD_ALIGN_LAT **0b1** The Common event is implemented.	x
[0]	IDhi0	IDhi0 corresponds to common event (0x4020) LDST_ALIGN_LAT **0b1** The Common event is implemented.	x

Access

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

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 IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.4.32 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

RESERVEDW

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-80: ext_pmsscr bit assignments

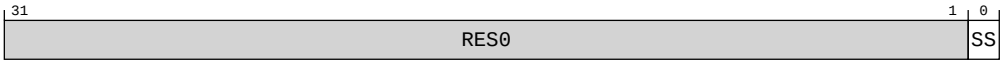


Table B-163: PMSSCR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	SS	Capture now. **0b0** Ignored. **0b1** Initiate a capture immediately.	x

Accessibility

Component	Offset
PMU	0xE30

This interface is accessible as follows:

WO

B.4.33 PMMIR, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE40

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx 0110 0000 0010 0000 0011


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-81: ext_pmmir bit assignments

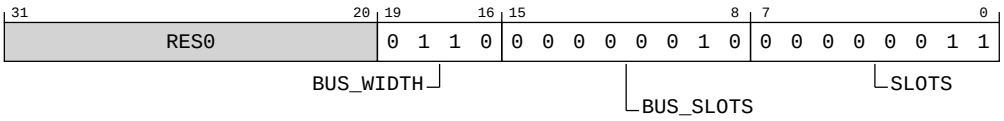


Table B-165: PMMIR bit descriptions

Bits	Name	Description	Reset
[31:20]	**RES0**	Reserved	**RES0**
[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. Defined values are: **0b0110** 32 bytes.	0b0110
[15:8]	BUS_SLOTS	Bus count. The largest value by which the BUS_ACCESS event might increment by in a single BUS_CYCLES cycle. **0b00000010** The largest value by which the BUS_ACCESS PMU event may increment in one cycle is 2.	0x02
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero. **0b00000011** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 3.	0x03

Accessibility

If the Core power domain is off or in a low-power state, access on this interface returns an Error.

Component	Offset	Instance	Range
PMU	0xE40	PMMIR	None

This interface is accessible as follows:

When !IsCorePowered() || DoubleLockStatus() || OSLockStatus() || !AllowExternalPMUAccess()
ERROR

Otherwise
RO

B.4.34 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

See bit descriptions

Reset value

0100 0111 0111 0000 0010 1010 0001 0110

Bit descriptions

Figure B-82: ext_pmdevarch bit assignments

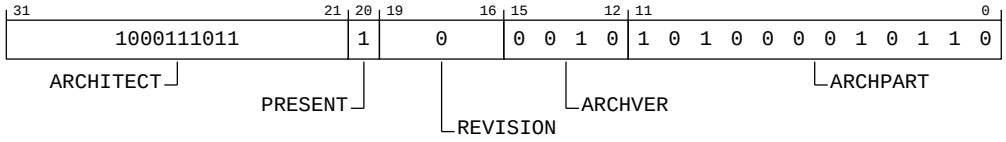


Table B-167: PMDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For Performance Monitors, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, `0x4`. Bits [27:21] are the JEP106 ID code, `0x3B`. **`0b01000111011`**	`0b01000111011`
[20]	PRESENT	Indicates that the DEVARCH 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`

Bits	Name	Description	Reset
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **0b0010** Performance Monitors Extension version 3, PMUv3. All other values are reserved. PMDEVARCH.ARCHVER and PMDEVARCH.ARCHPART are also defined as a single field, PMDEVARCH.ARCHID, so that PMDEVARCH.ARCHVER is PMDEVARCH.ARCHID[15:12].	0b0010
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **0b101000010110** Armv8-A PE performance monitors.	0xA16

Accessibility

Component	Offset	Instance	Range
PMU	0xFBC	PMDEVARCH	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.4.35 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.

Before Armv8.2, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of ext-EDDEVID.PCSample.

Attributes

Width

32

Component

PMU

Register offset

0xFC8

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-83: ext_pmdevid bit assignments

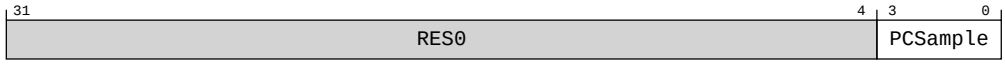


Table B-169: PMDEVID bit descriptions

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.	xxxx

Accessibility

Component	Offset	Instance	Range
PMU	0xFC8	PMDEVID	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.4.36 PMDEVTYPE, Performance Monitors Device Type register

Indicates to a debugger that this component is part of a PEs 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

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0001 0110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-84: ext_pmdevtype bit assignments

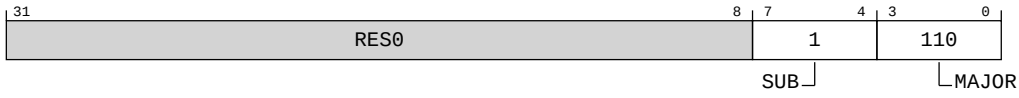


Table B-171: PMDEVTYPE bit descriptions

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()

RO

Otherwise

ERROR

B.4.37 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

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-85: ext_pmpidr4 bit assignments

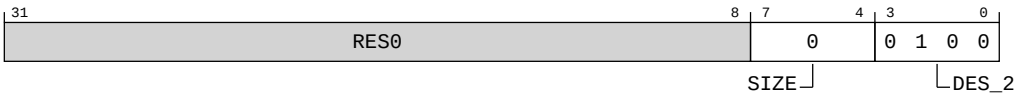


Table B-173: PMPIDR4 bit descriptions

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()

RO

Otherwise

ERROR

B.4.38 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

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1000 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-86: ext_pmpidr0 bit assignments

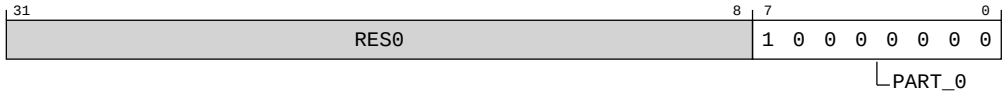


Table B-175: PMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b10000000** Cortex-A520	0x80

Accessibility

Component	Offset	Instance	Range
PMU	0xFE0	PMPIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.4.39 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

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-87: ext_pmpidr1 bit assignments

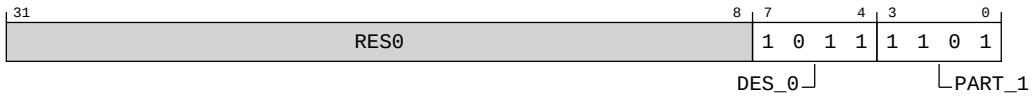


Table B-177: PMPIDR1 bit descriptions

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`** Cortex-A520	`0b1101`

Accessibility

Component	Offset	Instance	Range
PMU	0xFE4	PMPIDR1	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.4.40 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
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-88: ext_pmpidr2 bit assignments

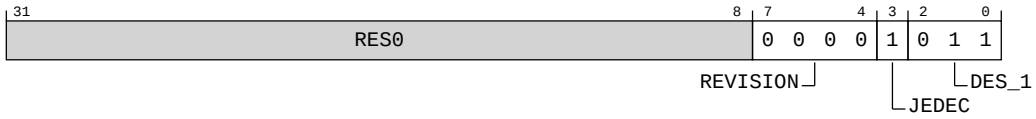


Table B-179: PMPIDR2 bit descriptions

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. **0b0000** r0p2	0b0000
[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()

RO

Otherwise

ERROR

B.4.41 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

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0010 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-89: ext_pmpidr3 bit assignments

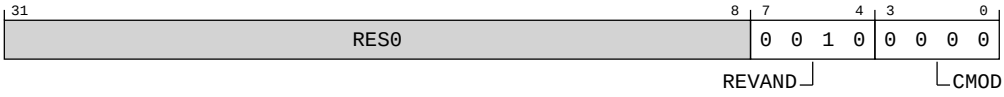


Table B-181: PMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	REVAND	Part minor revision. Parts using ext-PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0010** r0p2	0b0010
[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()
RO

Otherwise
ERROR

B.4.42 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
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-90: ext_pmcidr0 bit assignments

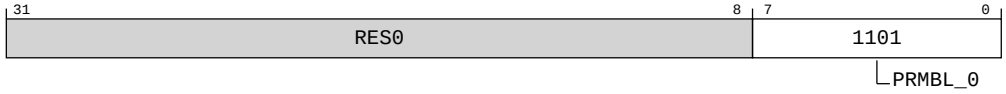


Table B-183: PMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble. **0b00001101**	0x0D

Accessibility

Component	Offset	Instance	Range
PMU	0xFF0	PMCIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.4.43 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

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-91: ext_pmcidr1 bit assignments

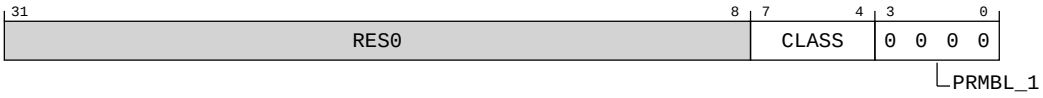


Table B-185: PMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight component. Other values are defined by the CoreSight Architecture. This field reads as 0x9.	xxxx
[3:0]	PRMBL_1	Preamble. **RAZ**. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
PMU	0xFF4	PMCIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.4.44 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

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-92: ext_pmcidr2 bit assignments

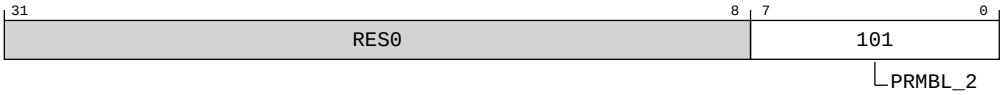


Table B-187: PMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble. **0b00000101**	0x05

Accessibility

Component	Offset	Instance	Range
PMU	0xFF8	PMCIDR2	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.4.45 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

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-93: ext_pmcidr3 bit assignments

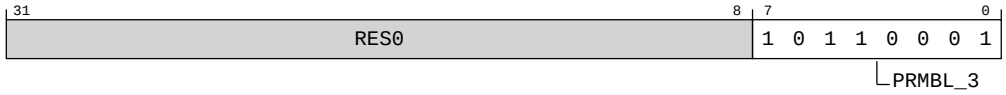


Table B-189: PMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble. **0b10110001**	0xB1

Accessibility

Component	Offset	Instance	Range
PMU	0xFFC	PMCIDR3	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5 External Debug registers summary

The 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-191: Debug registers summary

Offset	Name	Reset	Width	Description
0x020	EDESR	–	32-bit	External Debug Event Status Register
0x024	EDECR	–	32-bit	External Debug Execution Control Register
0x030	EDWAR[31:0]	–	32-bit	External Debug Watchpoint Address Register
0x034	EDWAR[63:32]	–	32-bit	External Debug Watchpoint Address Register
0x080	DBGDTRRX_EL0	–	32-bit	Debug Data Transfer Register, Receive
0x084	EDITR	–	32-bit	External Debug Instruction Transfer Register
0x088	EDSCR	–	32-bit	External Debug Status and Control Register
0x08C	DBGDTRTX_EL0	–	32-bit	Debug Data Transfer Register, Transmit
0x090	EDRCR	–	32-bit	External Debug Reserve Control Register
0x098	EDECCR	–	32-bit	External Debug Exception Catch Control Register
0x300	OSLAR_EL1	–	32-bit	OS Lock Access Register
0x310	EDPRCR	–	32-bit	External Debug Power/Reset Control Register
0x314	EDPRSR	–	32-bit	External Debug Processor Status Register
0x400	DBGBVR0_EL1[63:0]	–	64-bit	Debug Breakpoint Value Registers
0x408	DBGBCR0_EL1	–	32-bit	Debug Breakpoint Control Registers
0x410	DBGBVR1_EL1[63:0]	–	64-bit	Debug Breakpoint Value Registers
0x418	DBGBCR1_EL1	–	32-bit	Debug Breakpoint Control Registers
0x420	DBGBVR2_EL1[63:0]	–	64-bit	Debug Breakpoint Value Registers
0x428	DBGBCR2_EL1	–	32-bit	Debug Breakpoint Control Registers
0x430	DBGBVR3_EL1[63:0]	–	64-bit	Debug Breakpoint Value Registers
0x438	DBGBCR3_EL1	–	32-bit	Debug Breakpoint Control Registers
0x440	DBGBVR4_EL1[63:0]	–	64-bit	Debug Breakpoint Value Registers
0x448	DBGBCR4_EL1	–	32-bit	Debug Breakpoint Control Registers
0x450	DBGBVR5_EL1[63:0]	–	64-bit	Debug Breakpoint Value Registers
0x458	DBGBCR5_EL1	–	32-bit	Debug Breakpoint Control Registers
0x800	DBGWVR0_EL1[63:0]	–	64-bit	Debug Watchpoint Value Registers
0x808	DBGWCR0_EL1	–	32-bit	Debug Watchpoint Control Registers
0x810	DBGWVR1_EL1[63:0]	–	64-bit	Debug Watchpoint Value Registers

Offset	Name	Reset	Width	Description
0x818	DBGWCR1_EL1	–	32-bit	Debug Watchpoint Control Registers
0x820	DBGWVR2_EL1[63:0]	–	64-bit	Debug Watchpoint Value Registers
0x828	DBGWCR2_EL1	–	32-bit	Debug Watchpoint Control Registers
0x830	DBGWVR3_EL1[63:0]	–	64-bit	Debug Watchpoint Value Registers
0x838	DBGWCR3_EL1	–	32-bit	Debug Watchpoint Control Registers
0xD00	MIDR_EL1	–	32-bit	Main ID Register
0xD20	EDPFR[31:0]	–	32-bit	External Debug Processor Feature Register
0xD24	EDPFR[63:32]	–	32-bit	External Debug Processor Feature Register
0xD28	EDDFR[31:0]	–	32-bit	External Debug Feature Register
0xD2C	EDDFR[63:32]	–	32-bit	External Debug Feature Register
0xD60	EDAA32PFR	–	64-bit	External Debug Auxiliary Processor Feature Register
0xFA0	DBGCLAIMSET_EL1	–	32-bit	Debug CLAIM Tag Set register
0xFA4	DBGCLAIMCLR_EL1	–	32-bit	Debug CLAIM Tag Clear register
0xFA8	EDDEVAFF0	–	32-bit	External Debug Device Affinity register 0
0xFAC	EDDEVAFF1	–	32-bit	External Debug Device Affinity register 1
0xFB0	EDLAR	–	32-bit	External Debug Lock Access Register
0xFB4	EDLSR	–	32-bit	External Debug Lock Status Register
0xFB8	DBGAUTHSTATUS_EL1	–	32-bit	Debug Authentication Status register
0xFBC	EDDEVARCH	–	32-bit	External Debug Device Architecture register
0xFC0	EDDEVID2	–	32-bit	External Debug Device ID register 2
0xFC4	EDDEVID1	–	32-bit	External Debug Device ID register 1
0xFC8	EDDEVID	–	32-bit	External Debug Device ID register 0
0xFCC	EDDEVTYPE	–	32-bit	External Debug Device Type register
0xFD0	EDPIDR4	–	32-bit	External Debug Peripheral Identification Register 4
0xFE0	EDPIDR0	–	32-bit	External Debug Peripheral Identification Register 0
0xFE4	EDPIDR1	–	32-bit	External Debug Peripheral Identification Register 1
0xFE8	EDPIDR2	–	32-bit	External Debug Peripheral Identification Register 2
0xFEC	EDPIDR3	–	32-bit	External Debug Peripheral Identification Register 3
0xFF0	EDCIDR0	–	32-bit	External Debug Component Identification Register 0
0xFF4	EDCIDR1	–	32-bit	External Debug Component Identification Register 1
0xFF8	EDCIDR2	–	32-bit	External Debug Component Identification Register 2
0xFFC	EDCIDR3	–	32-bit	External Debug Component Identification Register 3

B.5.1 EDRCR, External Debug Reserve Control Register

This register is used to allow imprecise entry to Debug state and clear sticky bits in ext-EDSCR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0x090

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-94: ext_edrcr bit assignments

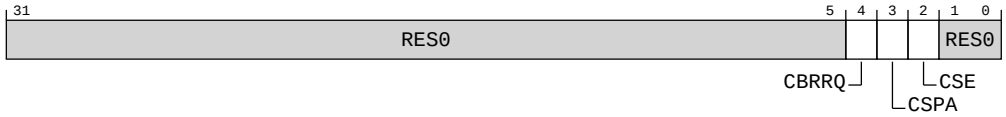


Table B-192: EDRCR bit descriptions

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. **0b1** Allow imprecise entry to Debug state, for example by canceling pending bus accesses. 'Setting this bit to 1 allows a debugger to request imprecise entry to Debug state. An External Debug Request debug event must be pending before the debugger sets this bit to 1. This feature is optional and implemented on ' + $Name + '.'	x

Bits	Name	Description	Reset
[3]	CSPA	Clear Sticky Pipeline Advance. This bit is used to clear the ext-EDSCR.PipeAdv bit to 0. The actions on writing to this bit are: **0b0** No action. **0b1** Clear the ext-EDSCR.PipeAdv bit to 0.	x
[2]	CSE	Clear Sticky Error. Used to clear the ext-EDSCR cumulative error bits to 0. The actions on writing to this bit are: **0b0** No action. **0b1** Clear the ext-EDSCR.{TXU, RXO, ERR} bits, and, if the PE is in Debug state, the ext-EDSCR.ITO bit, to 0.	x
[1:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
Debug	0x090	EDRCR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus()
WO

Otherwise
ERROR

B.5.2 EDPRCR, External Debug Power/Reset Control Register

Controls the PE functionality related to powerup, reset, and powerdown.

Configurations

If FEAT_DoPD is implemented then all fields in this register are in the Core power domain.

CORENPDRQ is the only field that is mapped between the EDPRCR and DBGPRCR and DBGPRCR_EL1.

Attributes

Width
32

Component
Debug

Register offset
0x310

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx0x

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-95: ext_edprcr bit assignments

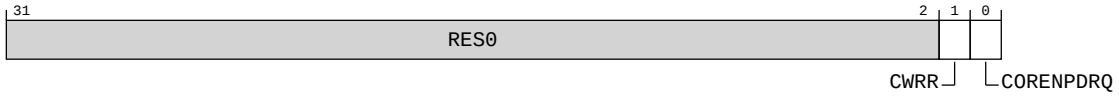


Table B-194: EDPRCR bit descriptions

Bits	Name	Description	Reset
[31:2]	**RES0**	Reserved	**RES0**
[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. This field is in the Core power domain The PE ignores writes to this bit if any of the following are true: • ExternalSecureInvasiveDebugEnabled() == FALSE and EL3 is implemented. In an implementation that includes the recommended external debug interface, this bit drives the DBGRSTREQ signal. **When OSLockStatus()** Access to this field is: **RAZ/WI** **Otherwise** Access to this field is: WO/**RAZ**	0b0

Bits	Name	Description	Reset
[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 powerup 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 power down the Core power domain, but instead emulates a powerdown of that domain. When this bit reads as **UNKNOWN**, the PE ignores writes to this bit. This field is in the Core power domain, and permitted accesses to this field map to the AArch32-DBGPRCR.CORENPDRQ and AArch64-DBGPRCR_EL1.CORENPDRQ fields. In an implementation that includes the recommended external debug interface, this bit drives the DBGNOPWRDWN signal. It is IMPLEMENTATION DEFINED whether this bit is reset to the Cold reset value on exit from an IMPLEMENTATION DEFINED software-visible retention state. For more information about retention states, see *Core power domain power states* in the *Arm® Architecture Reference Manual for A-profile architecture*. **Note:** Writes to this bit are not prohibited by the IMPLEMENTATION DEFINED authentication interface. This means that a debugger can request emulation of powerdown regardless of whether invasive debug is permitted. **When OSLockStatus()** Access to this field is: **UNKNOWN**/WI **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:

When IsCorePowered()

RW

Otherwise

ERROR

[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.

B.5.3 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 270 bits [31:0].

Attributes

Width

32

Component

Debug

Register offset

0xD00

Access type

See bit descriptions

Reset value

0100 0001 0000 1111 1101 1000 0000 0010

Bit descriptions

Figure B-96: ext_midr_el1 bit assignments

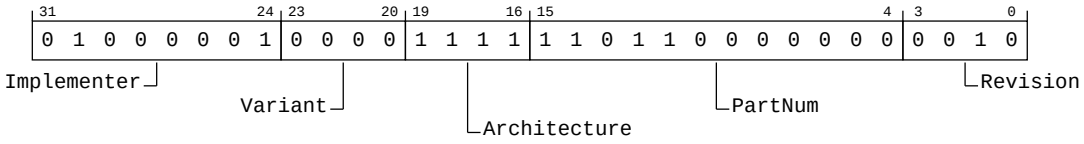


Table B-196: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	Implementer	Indicates the implementer code. This value is: **0b01000001** Arm Limited	0x41
[23:20]	Variant	Indicates the major revision of the product. **0b0000** r0p2	0b0000
[19:16]	Architecture	Architecture version. Defined values are: **0b1111** Architecture is defined by ID registers	0b1111

Bits	Name	Description	Reset
[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. **`0b110110000000`** Cortex-A520	`0xD80`
[3:0]	Revision	Indicates the minor revision of the product. **`0b0010`** r0p2	`0b0010`

Accessibility

Component	Offset	Instance	Range
Debug	0xD00	MIDR_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()
RO

Otherwise
ERROR

B.5.4 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 offsets (2)
0xD20,0xD24

Access type
See bit descriptions

Reset value

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

Figure B-97: ext_edpfr bit assignments

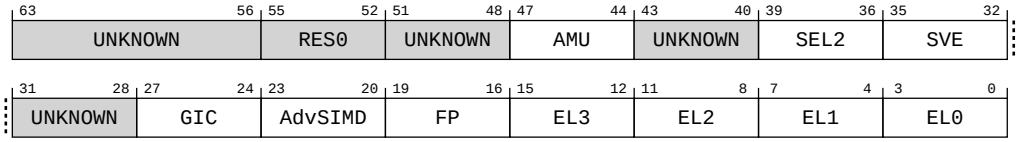


Table B-198: EDPFR bit descriptions

Bits	Name	Description	Reset
[63:56]	UNKNOWN	Reserved	UNKNOWN
[55:52]	RES0	Reserved	RES0
[51:48]	UNKNOWN	Reserved	UNKNOWN
[47:44]	AMU	Indicates support for Activity Monitors Extension. Defined values are: **0b0001** FEAT_AMUv1 is implemented.	xxxx
[43:40]	UNKNOWN	Reserved	UNKNOWN
[39:36]	SEL2	Secure EL2. Defined values are: **0b0001** Secure EL2 is implemented.	xxxx
[35:32]	SVE	Scalable Vector Extension. Defined values are: **0b0001** SVE is implemented.	xxxx
[31:28]	UNKNOWN	Reserved	UNKNOWN
[27:24]	GIC	System register GIC interface support. Defined values are: **0b0011** System register interface to version 4.1 of the GIC CPU interface is supported.	xxxx
[23:20]	AdvSIMD	Advanced SIMD. Defined values are: **0b0001** Advanced SIMD is implemented, including support for the following SISD and SIMD operations: • Integer byte, halfword, word and doubleword element operations. • Half-precision, single-precision and double-precision floating-point arithmetic. • Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.	xxxx

Bits	Name	Description	Reset
[19:16]	FP	Floating-point. Defined values are: **0b0001** Floating-point is implemented, and includes support for: • Half-precision, single-precision and double-precision floating-point types. • Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.	xxxx
[15:12]	EL3	AArch64 EL3 Exception level handling. Defined values are: **0b0001** EL3 can be executed in AArch64 state only.	xxxx
[11:8]	EL2	AArch64 EL2 Exception level handling. Defined values are: **0b0001** EL2 can be executed in AArch64 state only.	xxxx
[7:4]	EL1	AArch64 EL1 Exception level handling. Defined values are: **0b0001** EL1 can be executed in AArch64 state only.	xxxx
[3:0]	EL0	AArch64 EL0 Exception level handling. Defined values are: **0b0000** EL0 cannot be executed in AArch64 state. **0b0001** EL0 can be executed in AArch64 state only. **0b0010** EL0 can be executed in both Execution states. All other values are reserved. In an Armv8-A implementation that supports AArch64, this field returns the value of AArch64-ID_AA64PFR0_EL1.EL0.	xxxx

Accessibility

Component	Offset	Instance	Range
Debug	0xD20	EDPFR	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ERROR

Component	Offset	Instance	Range
Debug	0xD24	EDPFR	63:32

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ERROR

B.5.5 EDDFR, External Debug Feature Register

Provides top level information about the debug system.

Debuggers must use ext-EDDEVARCH to determine the Debug architecture version.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offsets (2)

0xD28,0xD2C

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0001 xxxx 0011 xxxx 0101 xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-98: ext_eddfr bit assignments

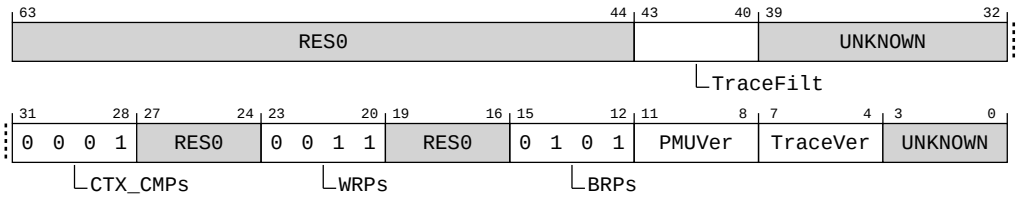


Table B-201: EDDFR bit descriptions

Bits	Name	Description	Reset
[63:44]	**RES0**	Reserved	**RES0**
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. Defined values are: **0b0001** Armv8.4 Self-hosted Trace Extension is implemented.	xxxx
[39:32]	**UNKNOWN**	Reserved	**UNKNOWN**
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. In an Armv8-A implementation that supports AArch64, this field returns the value of AArch64-ID_AA64DFR0_EL1.CTX_CMPs. **0b0001** Two context-aware breakpoints are included	0b0001
[27:24]	**RES0**	Reserved	**RES0**
[23:20]	WRPs	Number of watchpoints, minus 1. The value of 0b0000 is reserved. In an Armv8-A implementation that supports AArch64, this field returns the value of AArch64-ID_AA64DFR0_EL1.WRPs. **0b0011** Four watchpoints	0b0011
[19:16]	**RES0**	Reserved	**RES0**
[15:12]	BRPs	Number of breakpoints, minus 1. The value of 0b0000 is reserved. In an Armv8-A implementation that supports AArch64, this field returns the value of AArch64-ID_AA64DFR0_EL1.BRPs. **0b0101** Six breakpoints	0b0101
[11:8]	PMUVer	Performance Monitors Extension version. Defined value is: **0b0111** Performance Monitors Extension implemented, PMUv3 for Armv8.7	xxxx
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a PE trace unit is implemented. Defined values are: **0b0001** PE trace unit System registers implemented.	xxxx
[3:0]	**UNKNOWN**	Reserved	**UNKNOWN**

Accessibility

Component	Offset	Instance	Range
Debug	0xD28	EDDFR	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ERROR

Component	Offset	Instance	Range
Debug	0xD2C	EDDFR	63:32

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ERROR

B.5.6 EDDEVARCH, External Debug Device Architecture register

Identifies the programmers' model architecture of the external debug 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

Debug

Register offset

0xFBC

Access type

See bit descriptions

Reset value

0100 0111 0111 0000 xxxx 1010 0001 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-99: ext_eddevarch bit assignments

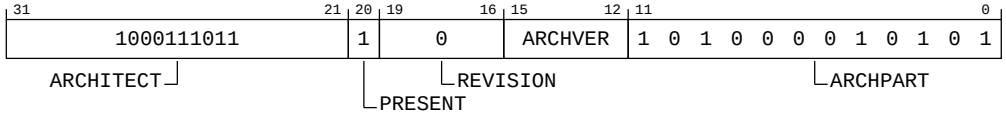


Table B-204: EDDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For debug, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, `0x4`. Bits [27:21] are the JEP106 ID code, `0x3B`. **`0b01000111011`**	`0b01000111011`
[20]	PRESENT	Indicates that the DEVARCH 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: **`0b1001`** Armv8.4 debug architecture, FEAT_Debugv8p4.	`xxxx`
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **`0b101000010101`** Armv8-A debug architecture.	`0xA15`

Accessibility

Component	Offset	Instance	Range
Debug	0xFBC	EDDEVARCH	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.7 EDDEVID2, External Debug Device ID register 2

Reserved for future descriptions of features of the debug 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.

Attributes

Width

32

Component

Debug

Register offset

0xFC0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-100: ext_eddevid2 bit assignments

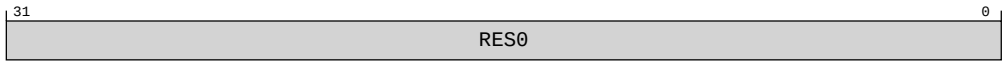


Table B-206: EDDEVID2 bit descriptions

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()

RO

Otherwise

ERROR

B.5.8 EDDEVID1, External Debug Device ID register 1

Provides extra information for external debuggers about features of the debug 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.

Attributes

Width

32

Component

Debug

Register offset

0xFC4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-101: ext_eddevid1 bit assignments

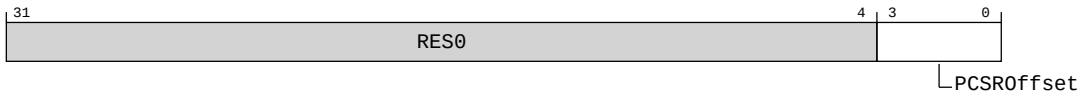


Table B-208: EDDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	PCSROffset	This field indicates the offset applied to PC samples returned by reads of ext-EDPCSR. Permitted values of this field in Armv8 are: **0b0000** ext-EDPCSR not implemented.	xxxx

Accessibility

Component	Offset	Instance	Range
Debug	0xFC4	EDDEVID1	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.5.9 EDDEVID, External Debug Device ID register 0

Provides extra information for external debuggers about features of the debug 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.

Attributes

Width
32

Component
Debug

Register offset
0xFC8

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-102: ext_eddevid bit assignments

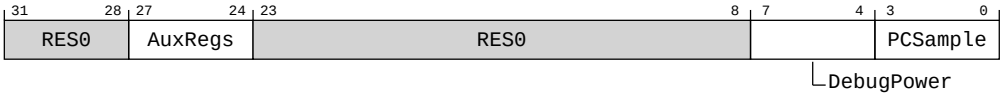


Table B-210: EDDEVID bit descriptions

Bits	Name	Description	Reset
[31:28]	RES0	Reserved	RES0
[27:24]	AuxRegs	Indicates support for Auxiliary registers. Defined values are: **0b0000** None supported.	xxxx
[23:8]	RES0	Reserved	RES0
[7:4]	DebugPower	Indicates support for the FEAT_DoPD feature. Defined values are: **0b0001** FEAT_DoPD implemented. All registers in the external debug interface register map are implemented in the Core power domain.	xxxx
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using external debug registers. Defined values are: **0b0000** PC Sample-based Profiling Extension is not implemented in the external debug registers space.	xxxx

Accessibility

Component	Offset	Instance	Range
Debug	0xFC8	EDDEVID	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.10 EDDEVTYPE, External Debug Device Type register

Indicates to a debugger that this component is part of a PEs debug logic.

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

Debug

Register offset

0xFCC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0001 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-103: ext_eddevtype bit assignments

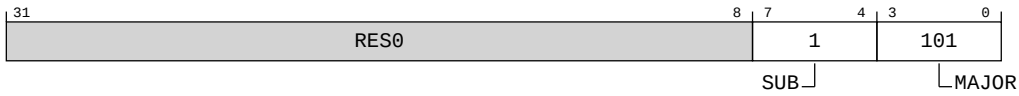


Table B-212: EDDEVTYPE bit descriptions

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()

RO

Otherwise

ERROR

B.5.11 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

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

Debug

Register offset

0xFD0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-104: ext_edpidr4 bit assignments

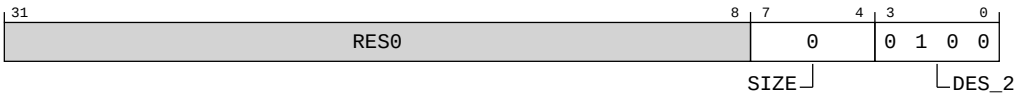


Table B-214: EDPIDR4 bit descriptions

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()

RO

Otherwise

ERROR

B.5.12 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

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

Debug

Register offset

0xFE0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1000 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-105: ext_edpidr0 bit assignments

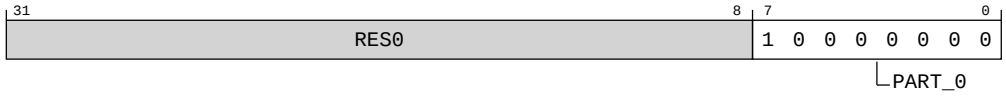


Table B-216: EDPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b10000000** Cortex-A520	0x80

Accessibility

Component	Offset	Instance	Range
Debug	0xFE0	EDPIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.13 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

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

Debug

Register offset

0xFE4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-106: ext_edpidr1 bit assignments

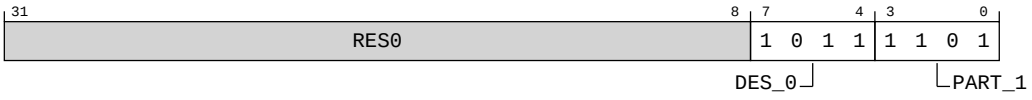


Table B-218: EDPIDR1 bit descriptions

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`** Cortex-A520	`0b1101`

Accessibility

Component	Offset	Instance	Range
Debug	0xFE4	EDPIDR1	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.5.14 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

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
Debug

Register offset
0xFE8

Access type
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-107: ext_edpidr2 bit assignments

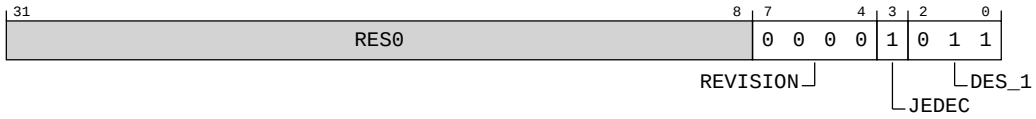


Table B-220: EDPIDR2 bit descriptions

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. **0b0000** r0p2	0b0000
[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()

RO

Otherwise

ERROR

B.5.15 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

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

Debug

Register offset

0xFEC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0010 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-108: ext_edpidr3 bit assignments

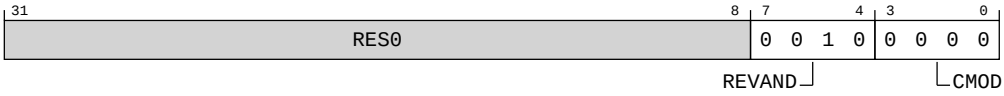


Table B-222: EDPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	REVAND	Part minor revision. Parts using ext-EDPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0010** r0p2	0b0010
[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()
RO

Otherwise
ERROR

B.5.16 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

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
Debug

Register offset
0xFF0

Access type
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-109: ext_edcidr0 bit assignments

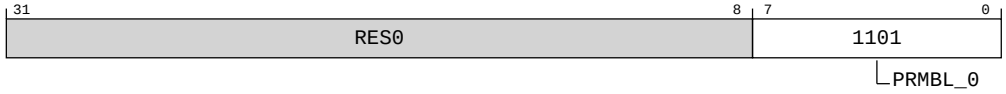


Table B-224: EDCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble. **0b00001101**	0x0D

Accessibility

Component	Offset	Instance	Range
Debug	0xFF0	EDCIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.17 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

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

Debug

Register offset

0xFF4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-110: ext_edcidr1 bit assignments

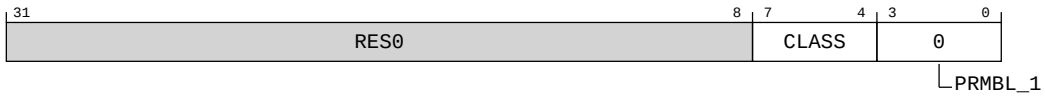


Table B-226: EDCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight component. Other values are defined by the CoreSight Architecture. This field reads as 0x9.	xxxx
[3:0]	PRMBL_1	Preamble. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFF4	EDCIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.18 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

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

Debug

Register offset

0xFF8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-111: ext_edcidr2 bit assignments

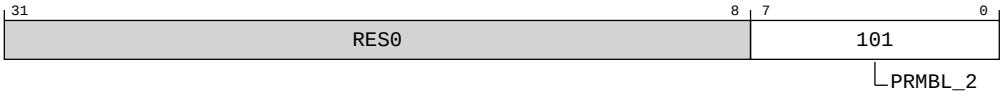


Table B-228: EDCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble. **0b00000101**	0x05

Accessibility

Component	Offset	Instance	Range
Debug	0xFF8	EDCIDR2	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.19 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

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

Debug

Register offset

0xFFC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-112: ext_edcidr3 bit assignments

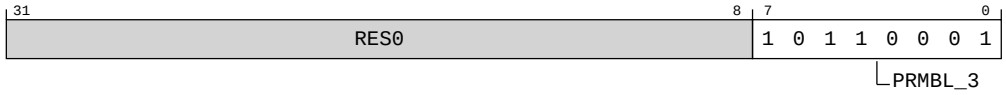


Table B-230: EDCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble. **0b10110001**	0xB1

Accessibility

Component	Offset	Instance	Range
Debug	0xFFC	EDCIDR3	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.6 External AMU registers summary

The 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-232: AMU registers summary

Offset	Name	Reset	Width	Description
0x0	AMEVCNTR00[31:0]	–	32-bit	Activity Monitors Event Counter Registers 0
0x4	AMEVCNTR00[63:32]	–	32-bit	Activity Monitors Event Counter Registers 0
0x8	AMEVCNTR01[31:0]	–	32-bit	Activity Monitors Event Counter Registers 0
0xC	AMEVCNTR01[63:32]	–	32-bit	Activity Monitors Event Counter Registers 0
0x10	AMEVCNTR02[31:0]	–	32-bit	Activity Monitors Event Counter Registers 0
0x14	AMEVCNTR02[63:32]	–	32-bit	Activity Monitors Event Counter Registers 0
0x18	AMEVCNTR03[31:0]	–	32-bit	Activity Monitors Event Counter Registers 0
0x1C	AMEVCNTR03[63:32]	–	32-bit	Activity Monitors Event Counter Registers 0
0x100	AMEVCNTR10[31:0]	–	32-bit	Activity Monitors Event Counter Registers 1
0x104	AMEVCNTR10[63:32]	–	32-bit	Activity Monitors Event Counter Registers 1
0x108	AMEVCNTR11[31:0]	–	32-bit	Activity Monitors Event Counter Registers 1
0x10C	AMEVCNTR11[63:32]	–	32-bit	Activity Monitors Event Counter Registers 1
0x110	AMEVCNTR12[31:0]	–	32-bit	Activity Monitors Event Counter Registers 1
0x114	AMEVCNTR12[63:32]	–	32-bit	Activity Monitors Event Counter Registers 1
0x400	AMEVTYPER00	–	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	–	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	–	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	–	32-bit	Activity Monitors Event Type Registers 0
0x480	AMEVTYPER10	–	32-bit	Activity Monitors Event Type Registers 1
0x484	AMEVTYPER11	–	32-bit	Activity Monitors Event Type Registers 1
0x488	AMEVTYPER12	–	32-bit	Activity Monitors Event Type Registers 1
0xC00	AMCNTENSET0	–	32-bit	Activity Monitors Count Enable Set Register 0
0xC04	AMCNTENSET1	–	32-bit	Activity Monitors Count Enable Set Register 1
0xC20	AMCNTENCLR0	–	32-bit	Activity Monitors Count Enable Clear Register 0
0xC24	AMCNTENCLR1	–	32-bit	Activity Monitors Count Enable Clear Register 1
0xCE0	AMCGCR	–	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	–	32-bit	Activity Monitors Configuration Register
0xE04	AMCR	–	32-bit	Activity Monitors Control Register

Offset	Name	Reset	Width	Description
0xE08	AMIIDR	—	32-bit	Activity Monitors Implementation Identification Register
0xFA8	AMDEVAFF0	—	32-bit	Activity Monitors Device Affinity Register 0
0xFAC	AMDEVAFF1	—	32-bit	Activity Monitors Device Affinity Register 1
0xFBC	AMDEVARCH	—	32-bit	Activity Monitors Device Architecture Register
0xFCC	AMDEVTYPE	—	32-bit	Activity Monitors Device Type Register
0xFD0	AMPIDR4	—	32-bit	Activity Monitors Peripheral Identification Register 4
0xFE0	AMPIDR0	—	32-bit	Activity Monitors Peripheral Identification Register 0
0xFE4	AMPIDR1	—	32-bit	Activity Monitors Peripheral Identification Register 1
0xFE8	AMPIDR2	—	32-bit	Activity Monitors Peripheral Identification Register 2
0xFEC	AMPIDR3	—	32-bit	Activity Monitors Peripheral Identification Register 3
0xFF0	AMCIDR0	—	32-bit	Activity Monitors Component Identification Register 0
0xFF4	AMCIDR1	—	32-bit	Activity Monitors Component Identification Register 1
0xFF8	AMCIDR2	—	32-bit	Activity Monitors Component Identification Register 2
0xFFC	AMCIDR3	—	32-bit	Activity Monitors Component Identification Register 3

B.6.1 AMEVTYPER00, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR00_EL0 counts.

Configurations

External register AMEVTYPER00 bits [31:0] are architecturally mapped to AArch64 System register A.11.3 AMEVTYPER00_EL0, Activity Monitors Event Type Registers 0 on page 390 bits [31:0].

Attributes

Width

32

Component

AMU

Register offset

0x400

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-113: ext_amevtyper00 bit assignments

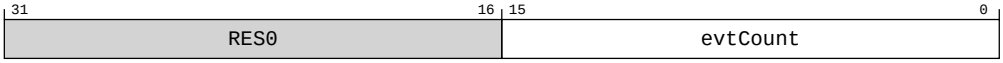


Table B-233: AMEVTYPER00 bit descriptions

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 ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0000000000010001** Processor frequency cycles	16{x}

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> 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

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-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.6.2 AMEVTYPER01, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR01_EL0 counts.

Configurations

External register AMEVTYPER01 bits [31:0] are architecturally mapped to AArch64 System register A.11.4 AMEVTYPER01_EL0, Activity Monitors Event Type Registers 0 on page 392 bits [31:0].

Attributes

Width

32

Component

AMU

Register offset

0x404

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-114: ext_amevtyper01 bit assignments

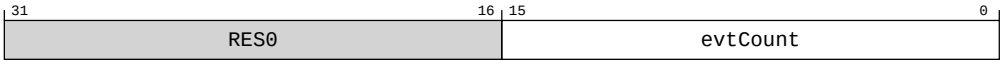


Table B-235: AMEVTYPER01 bit descriptions

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 ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0100000000000100** Constant frequency cycles	16{x}

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> 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

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-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.6.3 AMEVTYPER02, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR02_EL0 counts.

Configurations

External register AMEVTYPER02 bits [31:0] are architecturally mapped to AArch64 System register A.11.5 AMEVTYPER02_EL0, Activity Monitors Event Type Registers 0 on page 394 bits [31:0].

Attributes

Width

32

Component

AMU

Register offset

0x408

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-115: ext_amevtyper02 bit assignments

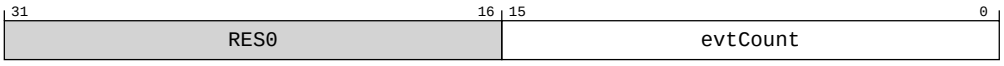


Table B-237: AMEVTYPER02 bit descriptions

Bits	Name	Description	Reset
[31:16]	**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 ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0000000000001000** Instructions retired	16{x}

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> 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

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-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.6.4 AMEVTYPER03, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR03_EL0 counts.

Configurations

External register AMEVTYPER03 bits [31:0] are architecturally mapped to AArch64 System register A.11.6 AMEVTYPER03_EL0, Activity Monitors Event Type Registers 0 on page 397 bits [31:0].

Attributes

Width

32

Component

AMU

Register offset

0x40C

Access type

Read

R

Write

RESERVED

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-116: ext_amevtyper03 bit assignments

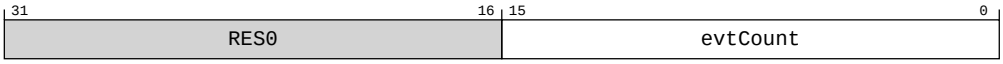


Table B-239: AMEVTYPER03 bit descriptions

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 ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0100000000000101** Memory stall cycles	16{x}

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> 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

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-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.6.5 AMEVTYPER10, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR10_EL0 counts.

Configurations

External register AMEVTYPER10 bits [31:0] are architecturally mapped to AArch64 System register A.11.7 AMEVTYPER10_EL0, Activity Monitors Event Type Registers 1 on page 399 bits [31:0].

Attributes

Width

32

Component

AMU

Register offset

0x480

Access type

Read

R

Write

RESERVED

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-117: ext_amevtyper10 bit assignments

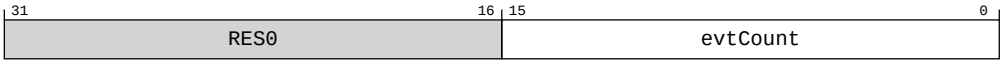


Table B-241: AMEVTYPER10 bit descriptions

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 AMEVCNTR10_EL0. **0b0000001100000000** MPMM gear 0 period threshold exceeded	16{x}

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> 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

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.


Note

ext-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.6.6 AMEVTYPER11, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR11_EL0 counts.

Configurations

External register AMEVTYPER11 bits [31:0] are architecturally mapped to AArch64 System register A.11.8 AMEVTYPER11_EL0, Activity Monitors Event Type Registers 1 on page 401 bits [31:0].

Attributes

Width

32

Component

AMU

Register offset

0x484

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-118: ext_amevtyper11 bit assignments

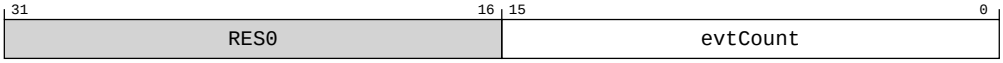


Table B-243: AMEVTYPER11 bit descriptions

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 AMEVCNTR11_EL0. **0b0000001100000001** MPMM gear 1 period threshold exceeded	16{x}

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> 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

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-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.6.7 AMEVTYPER12, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR12_EL0 counts.

Configurations

External register AMEVTYPER12 bits [31:0] are architecturally mapped to AArch64 System register A.11.9 AMEVTYPER12_EL0, Activity Monitors Event Type Registers 1 on page 403 bits [31:0].

Attributes

Width

32

Component

AMU

Register offset

0x488

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-119: ext_amevtyper12 bit assignments

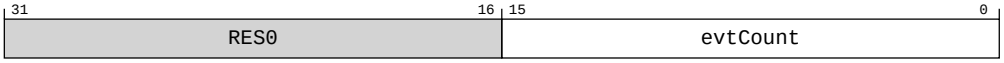


Table B-245: AMEVTYPER12 bit descriptions

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 AMEVCNTR12_EL0. **0b0000001100000010** MPMM gear 2 period threshold exceeded	16{x}

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> 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

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-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.6.8 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.11.2 AMCGCR_EL0, Activity Monitors Counter Group Configuration Register on page 388 bits [31:0].

Attributes

Width

32

Component

AMU

Register offset

0xCE0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx 0000 0011 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-120: ext_amcgcr bit assignments

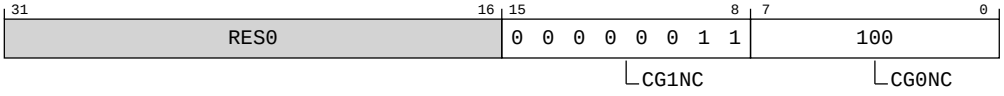


Table B-247: AMCGCR bit descriptions

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. **0b00000011** Three counters in the auxiliary counter group	0x03

Bits	Name	Description	Reset
[7:0]	CG0NC	Counter Group 0 Number of Counters. The number of counters in the architected counter group. **0b00000100**	0x04

Accessibility

Component	Offset	Instance	Range
AMU	0xCE0	AMCGCR	None

This interface is accessible as follows:

RO

B.6.9 AMCFGR, Activity Monitors Configuration Register

Global configuration register for the activity monitors.

Provides information on supported features, the number of counter groups implemented, the total number of activity monitor event counters implemented, and the size of the counters. AMCFGR is applicable to both the architected and the auxiliary counter groups.

Configurations

External register AMCFGR bits [31:0] are architecturally mapped to AArch64 System register A.11.1 AMCFGR_EL0, Activity Monitors Configuration Register on page 385 bits [31:0].

Attributes

Width

32

Component

AMU

Register offset

0xE00

Access type

RO

Reset value

0001 xxx1 0000 0000 0011 1111 0000 0110

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-121: ext_amcfgr bit assignments

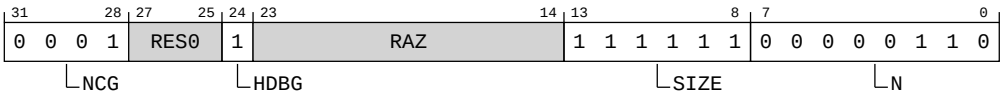


Table B-249: AMCFGR bit descriptions

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** ext-AMCR.HDBG is read/write.	0b1
[23:14]	**RAZ**	Reserved	**RAZ**
[13:8]	SIZE	Defines the size of activity monitor event counters. The size of the activity monitor event counters implemented by the Activity Monitors Extension is [AMCFGR.SIZE + 1]. The counters are 64-bit. **Note:** Software also uses this field to determine the spacing of 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. The total number of counters implemented in all groups by the Activity Monitors Extension is [AMCFGR.N + 1]. **0b00000110** Seven activity monitor event counters	0x06

Accessibility

Component	Offset	Instance	Range
AMU	0xE00	AMCFGR	None

This interface is accessible as follows:

RO

B.6.10 AMIIDR, Activity Monitors Implementation Identification Register

Defines the implementer and revisions of the AMU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xE08

Access type

RO

Reset value

1101 1000 0000 0000 0010 0100 0011 1011

Bit descriptions

Figure B-122: ext_amiidr bit assignments

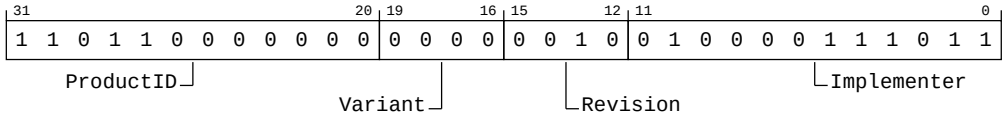


Table B-251: AMIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	This field is an AMU part identifier. **0b110110000000** Cortex-A520	0xD80
[19:16]	Variant	This field distinguishes product variants or major revisions of the product. **0b0000** r0p2	0b0000
[15:12]	Revision	This field distinguishes minor revisions of the product. **0b0010** r0p2	0b0010

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`. **`0b010000111011`** Arm Limited	`0x43B`

Accessibility

Component	Offset	Instance	Range
AMU	0xE08	AMIIDR	None

This interface is accessible as follows:

RO

B.6.11 AMDEVARCH, Activity Monitors Device Architecture Register

Identifies the programmers' model architecture of the AMU component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFBC

Access type

RO

Reset value

0100 0111 0111 0000 0000 1010 0110 0110

Bit descriptions

Figure B-123: ext_amdevarch bit assignments

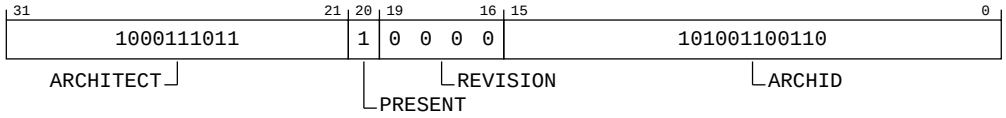


Table B-253: AMDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For AMU, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, `0x4`. Bits [27:21] are the JEP106 ID code, `0x3B`. **`0b01000111011`**	`0b01000111011`
[20]	PRESENT	Indicates that the DEVARCH is present. **`0b1`**	`0b1`
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. **`0b0000`** Architecture revision is AMUv1. All other values are reserved.	`0b0000`
[15:0]	ARCHID	Defines this part to be an AMU component. For architectures defined by Arm this is further subdivided. For AMU: • Bits [15:12] are the architecture version, `0x0`. • Bits [11:0] are the architecture part number, `0xA66`. This corresponds to AMU architecture version AMUv1. **`0b0000101001100110`**	`0x0A66`

Accessibility

Component	Offset	Instance	Range
AMU	0xFBC	AMDEVARCH	None

This interface is accessible as follows:

RO

B.6.12 AMDEVTYPE, Activity Monitors Device Type Register

Indicates to a debugger that this component is part of a PE's performance monitor interface.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFCC

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0001 0110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-124: ext_amdevtype bit assignments

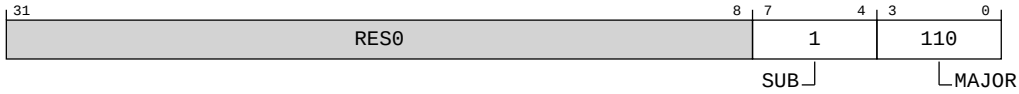


Table B-255: AMDEVTYPE bit descriptions

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.6.13 AMPIDR4, Activity Monitors Peripheral Identification Register 4

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual 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

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-125: ext_ampidr4 bit assignments

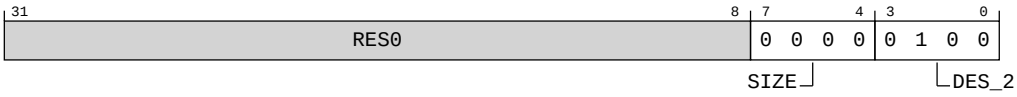


Table B-257: AMPIDR4 bit descriptions

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

Bits	Name	Description	Reset
[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.6.14 AMPIDR0, Activity Monitors Peripheral Identification Register 0

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* 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

xxxx xxxx xxxx xxxx xxxx xxxx 1000 0000

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-126: ext_ampidr0 bit assignments

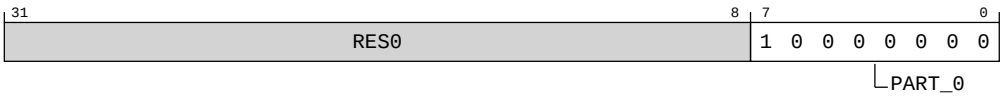


Table B-259: AMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b10000000** Cortex-A520	0x80

Accessibility

Component	Offset	Instance	Range
AMU	0xFE0	AMPIDR0	None

This interface is accessible as follows:

RO

B.6.15 AMPIDR1, Activity Monitors Peripheral Identification Register 1

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* 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

xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-127: ext_ampidr1 bit assignments

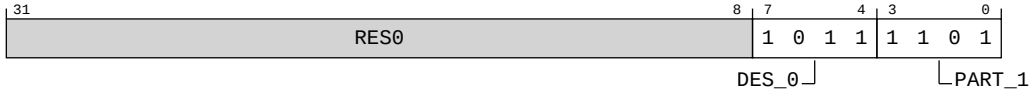


Table B-261: AMPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is `0b1011`. **`0b1011`** Arm Limited	`0b1011`
[3:0]	PART_1	Part number, most significant nibble. **`0b1101`** Cortex-A520	`0b1101`

Accessibility

Component	Offset	Instance	Range
AMU	0xFE4	AMPIDR1	None

This interface is accessible as follows:

RO

B.6.16 AMPIDR2, Activity Monitors Peripheral Identification Register 2

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* 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

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-128: ext_ampidr2 bit assignments

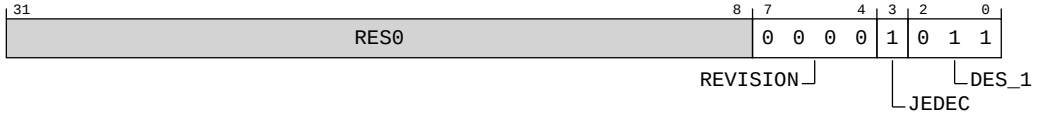


Table B-263: AMPIDR2 bit descriptions

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. **0b0000** r0p2	0b0000
[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.6.17 AMPIDR3, Activity Monitors Peripheral Identification Register 3

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual 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

xxxx xxxx xxxx xxxx xxxx xxxx 0010 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-129: ext_ampidr3 bit assignments

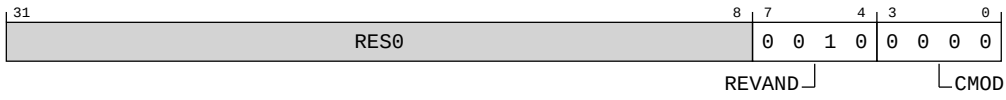


Table B-265: AMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Part minor revision. Parts using ext-AMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0010** r0p2	0b0010
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
AMU	0xFEC	AMPIDR3	None

This interface is accessible as follows:

RO

B.6.18 AMCIDR0, Activity Monitors Component Identification Register 0

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual 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

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-130: ext_amcidr0 bit assignments

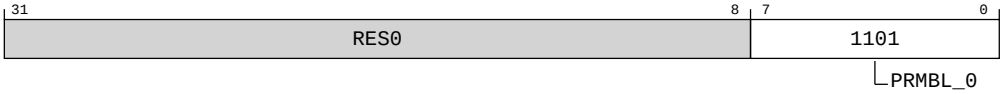


Table B-267: AMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble. **0b00001101**	0x0D

Accessibility

Component	Offset	Instance	Range
AMU	0xFF0	AMCIDR0	None

This interface is accessible as follows:

RO

B.6.19 AMCIDR1, Activity Monitors Component Identification Register 1

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual 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

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-131: ext_amcidr1 bit assignments

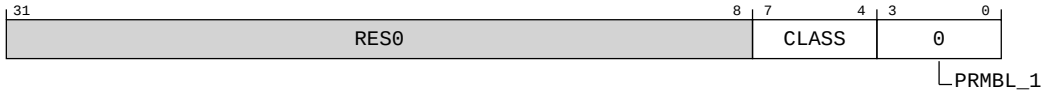


Table B-269: AMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight component. Other values are defined by the CoreSight Architecture. This field reads as 0x9.	xxxx
[3:0]	PRMBL_1	Preamble. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
AMU	0xFF4	AMCIDR1	None

This interface is accessible as follows:

RO

B.6.20 AMCIDR2, Activity Monitors Component Identification Register 2

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual 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

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-132: ext_amcidr2 bit assignments

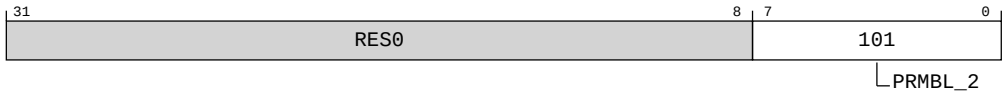


Table B-271: AMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble. **0b00000101**	0x05

Accessibility

Component	Offset	Instance	Range
AMU	0xFF8	AMCIDR2	None

This interface is accessible as follows:

RO

B.6.21 AMCIDR3, Activity Monitors Component Identification Register 3

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual 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

xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-133: ext_amcidr3 bit assignments

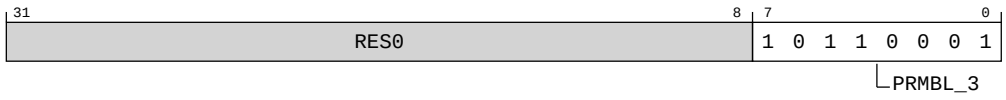


Table B-273: AMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble. **0b10110001**	0xB1

Accessibility

Component	Offset	Instance	Range
AMU	0xFFC	AMCIDR3	None

This interface is accessible as follows:

RO

B.7 External ETE registers summary

The 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 B-275: ETE registers summary

Offset	Name	Reset	Width	Description
0x004	TRCPRGCTLR	—	32-bit	Programming Control Register
0x00C	TRCSTATR	—	32-bit	Trace Status Register
0x010	TRCCONFIGR	—	32-bit	Trace Configuration Register
0x018	TRCAUXCTLR	—	32-bit	Auxiliary Control Register
0x020	TRCEVENTCTL0R	—	32-bit	Event Control 0 Register
0x024	TRCEVENTCTL1R	—	32-bit	Event Control 1 Register
0x028	TRCRSR	—	32-bit	Resources Status Register
0x02C	TRCSTALLCTLR	—	32-bit	Stall Control Register
0x030	TRCTSCTLR	—	32-bit	Timestamp Control Register
0x034	TRCSYNCPR	—	32-bit	Synchronization Period Register
0x038	TRCCCCTLR	—	32-bit	Cycle Count Control Register
0x03C	TRCBBCTLR	—	32-bit	Branch Broadcast Control Register
0x040	TRCTRACEIDR	—	32-bit	Trace ID Register

Offset	Name	Reset	Width	Description
0x044	TRCQCTLR	—	32-bit	Q Element Control Register
0x080	TRCVICTLR	—	32-bit	ViewInst Main Control Register
0x084	TRCVIIECTLR	—	32-bit	ViewInst Include/Exclude Control Register
0x088	TRCVISSCTLR	—	32-bit	ViewInst Start/Stop Control Register
0x100	TRCSEQEVR0	—	32-bit	Sequencer State Transition Control Register <n>
0x104	TRCSEQEVR1	—	32-bit	Sequencer State Transition Control Register <n>
0x108	TRCSEQEVR2	—	32-bit	Sequencer State Transition Control Register <n>
0x118	TRCSEQRSTEVR	—	32-bit	Sequencer Reset Control Register
0x11C	TRCSEQSTR	—	32-bit	Sequencer State Register
0x120	TRCEXTINSELR0	—	32-bit	External Input Select Register <n>
0x124	TRCEXTINSELR1	—	32-bit	External Input Select Register <n>
0x128	TRCEXTINSELR2	—	32-bit	External Input Select Register <n>
0x12C	TRCEXTINSELR3	—	32-bit	External Input Select Register <n>
0x140	TRCCNTRLDVR0	—	32-bit	Counter Reload Value Register <n>
0x144	TRCCNTRLDVR1	—	32-bit	Counter Reload Value Register <n>
0x150	TRCCNTCTLR0	—	32-bit	Counter Control Register <n>
0x154	TRCCNTCTLR1	—	32-bit	Counter Control Register <n>
0x160	TRCCNTVR0	—	32-bit	Counter Value Register <n>
0x164	TRCCNTVR1	—	32-bit	Counter Value Register <n>
0x180	TRCIDR8	—	32-bit	ID Register 8
0x184	TRCIDR9	—	32-bit	ID Register 9
0x188	TRCIDR10	—	32-bit	ID Register 10
0x18C	TRCIDR11	—	32-bit	ID Register 11
0x190	TRCIDR12	—	32-bit	ID Register 12
0x194	TRCIDR13	—	32-bit	ID Register 13
0x1C0	TRCIMSPEC0	—	32-bit	IMP DEF Register 0
0x1E0	TRCIDR0	—	32-bit	ID Register 0
0x1E4	TRCIDR1	—	32-bit	ID Register 1
0x1E8	TRCIDR2	—	32-bit	ID Register 2
0x1EC	TRCIDR3	—	32-bit	ID Register 3
0x1F0	TRCIDR4	—	32-bit	ID Register 4
0x1F4	TRCIDR5	—	32-bit	ID Register 5
0x1F8	TRCIDR6	—	32-bit	ID Register 6
0x1FC	TRCIDR7	—	32-bit	ID Register 7
0x208	TRCRSCTLR2	—	32-bit	Resource Selection Control Register <n>
0x20C	TRCRSCTLR3	—	32-bit	Resource Selection Control Register <n>
0x210	TRCRSCTLR4	—	32-bit	Resource Selection Control Register <n>
0x214	TRCRSCTLR5	—	32-bit	Resource Selection Control Register <n>
0x218	TRCRSCTLR6	—	32-bit	Resource Selection Control Register <n>
0x21C	TRCRSCTLR7	—	32-bit	Resource Selection Control Register <n>

Offset	Name	Reset	Width	Description
0x220	TRCRSCTLR8	—	32-bit	Resource Selection Control Register <n>
0x224	TRCRSCTLR9	—	32-bit	Resource Selection Control Register <n>
0x228	TRCRSCTLR10	—	32-bit	Resource Selection Control Register <n>
0x22C	TRCRSCTLR11	—	32-bit	Resource Selection Control Register <n>
0x230	TRCRSCTLR12	—	32-bit	Resource Selection Control Register <n>
0x234	TRCRSCTLR13	—	32-bit	Resource Selection Control Register <n>
0x238	TRCRSCTLR14	—	32-bit	Resource Selection Control Register <n>
0x23C	TRCRSCTLR15	—	32-bit	Resource Selection Control Register <n>
0x280	TRCSSCCR0	—	32-bit	Single-shot Comparator Control Register <n>
0x2A0	TRCSSCSR0	—	32-bit	Single-shot Comparator Control Status Register <n>
0x304	TRCOSLSR	—	32-bit	Trace OS Lock Status Register
0x310	TRCPDCR	—	32-bit	PowerDown Control Register
0x314	TRCPDSR	—	32-bit	PowerDown Status Register
0x400	TRCACVR0	—	64-bit	Address Comparator Value Register <n>
0x408	TRCACVR1	—	64-bit	Address Comparator Value Register <n>
0x410	TRCACVR2	—	64-bit	Address Comparator Value Register <n>
0x418	TRCACVR3	—	64-bit	Address Comparator Value Register <n>
0x420	TRCACVR4	—	64-bit	Address Comparator Value Register <n>
0x428	TRCACVR5	—	64-bit	Address Comparator Value Register <n>
0x430	TRCACVR6	—	64-bit	Address Comparator Value Register <n>
0x438	TRCACVR7	—	64-bit	Address Comparator Value Register <n>
0x480	TRCACATR0	—	64-bit	Address Comparator Access Type Register <n>
0x488	TRCACATR1	—	64-bit	Address Comparator Access Type Register <n>
0x490	TRCACATR2	—	64-bit	Address Comparator Access Type Register <n>
0x498	TRCACATR3	—	64-bit	Address Comparator Access Type Register <n>
0x4A0	TRCACATR4	—	64-bit	Address Comparator Access Type Register <n>
0x4A8	TRCACATR5	—	64-bit	Address Comparator Access Type Register <n>
0x4B0	TRCACATR6	—	64-bit	Address Comparator Access Type Register <n>
0x4B8	TRCACATR7	—	64-bit	Address Comparator Access Type Register <n>
0x600	TRCCIDCVR0	—	64-bit	Context Identifier Comparator Value Registers <n>
0x640	TRCVMIDCVR0	—	64-bit	Virtual Context Identifier Comparator Value Register <n>
0x680	TRCCIDCCTLR0	—	32-bit	Context Identifier Comparator Control Register 0
0x688	TRCVMIDCCTLR0	—	32-bit	Virtual Context Identifier Comparator Control Register 0
0xEE4	TRCITATBIDR	—	32-bit	Trace Intergration ATB Identification Register
0xEEC	TRCITATBDATAR	—	32-bit	Trace Integration Test ATB Data Register 0
0xEF4	TRCITATBINR	—	32-bit	Trace Integration ATB In Register
0xEFC	TRCITATBOUTR	—	32-bit	Trace Integration ATB Out Register
0xF00	TRCITCTRL	—	32-bit	Integration Mode Control Register
0xFA0	TRCCLAIMSET	—	32-bit	Claim Tag Set Register
0xFA4	TRCCLAIMCLR	—	32-bit	Claim Tag Clear Register

Offset	Name	Reset	Width	Description
0xFA8	TRCDEVAFF	—	64-bit	Device Affinity Register
0xFB0	TRCLAR	—	32-bit	Lock Access Register
0xFB4	TRCLSR	—	32-bit	Lock Status Register
0xFB8	TRCAUTHSTATUS	—	32-bit	Authentication Status Register
0xFBC	TRCDEVARCH	—	32-bit	Device Architecture Register
0xFC0	TRCDEVID2	—	32-bit	Device Configuration Register 2
0xFC4	TRCDEVID1	—	32-bit	Device Configuration Register 1
0xFC8	TRCDEVID	—	32-bit	Device Configuration Register
0xFCC	TRCDEVTYPE	—	32-bit	Device Type Register
0xFD0	TRCPIDR4	—	32-bit	Peripheral Identification Register 4
0xFD4	TRCPIDR5	—	32-bit	Peripheral Identification Register 5
0xFD8	TRCPIDR6	—	32-bit	Peripheral Identification Register 6
0xFDC	TRCPIDR7	—	32-bit	Peripheral Identification Register 7
0xFE0	TRCPIDR0	—	32-bit	Peripheral Identification Register 0
0xFE4	TRCPIDR1	—	32-bit	Peripheral Identification Register 1
0xFE8	TRCPIDR2	—	32-bit	Peripheral Identification Register 2
0xFEC	TRCPIDR3	—	32-bit	Peripheral Identification Register 3
0xFF0	TRCCIDR0	—	32-bit	Component Identification Register 0
0xFF4	TRCCIDR1	—	32-bit	Component Identification Register 1
0xFF8	TRCCIDR2	—	32-bit	Component Identification Register 2
0xFFC	TRCCIDR3	—	32-bit	Component Identification Register 3

B.7.1 TRCAUXCTLR, 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 A.13.8 TRCAUXCTLR, Auxiliary Control Register on page 425 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x018

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-134: ext_trcauxctlr bit assignments

Table B-276: 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() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.2 TRCIDR8, 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.13.1 TRCIDR8, ID Register 8 on page 413 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x180

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-135: ext_trcidr8 bit assignments

Table B-278: 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. **0b00000000000000000000000000000000**	32{x}

Accessibility

Component	Offset	Instance	Range
ETE	0x180	TRCIDR8	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.3 TRCIDR9, 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 A.13.3 TRCIDR9, ID Register 9 on page 417 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x184

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-136: ext_trcidr9 bit assignments

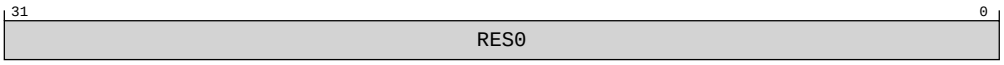


Table B-280: TRCIDR9 bit descriptions

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() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.4 TRCIDR10, 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.13.4 TRCIDR10, ID Register 10 on page 419 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x188

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-137: ext_trcidr10 bit assignments

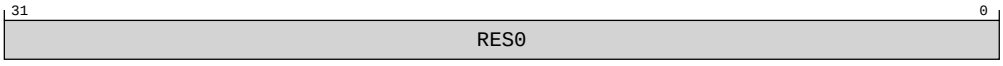


Table B-282: TRCIDR10 bit descriptions

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() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.5 TRCIDR11, 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.13.5 TRCIDR11, ID Register 11 on page 420 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x18C

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-138: ext_trcidr11 bit assignments

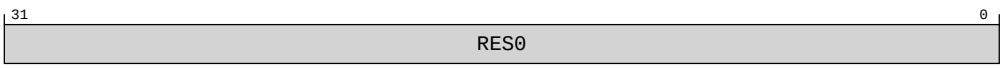


Table B-284: TRCIDR11 bit descriptions

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() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.6 TRCIDR12, 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.13.6 TRCIDR12, ID Register 12 on page 422 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x190

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-139: ext_trcidr12 bit assignments

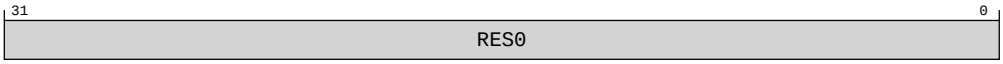


Table B-286: TRCIDR12 bit descriptions

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() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.7 TRCIDR13, 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.13.7 TRCIDR13, ID Register 13 on page 424 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x194

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-140: ext_trcidr13 bit assignments

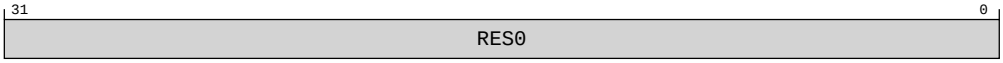


Table B-288: TRCIDR13 bit descriptions

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() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.8 TRCIMSPEC0, IMP DEF Register 0

TRCIMSPEC0 shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

Configurations

External register TRCIMSPEC0 bits [31:0] are architecturally mapped to AArch64 System register A.13.2 TRCIMSPEC0, IMP DEF Register 0 on page 415 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1C0

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-141: ext_trcimspec0 bit assignments

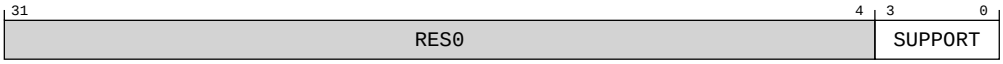


Table B-290: TRCIMSPEC0 bit descriptions

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.	xxxx

Accessibility

Component	Offset	Instance	Range
ETE	0x1C0	TRCIMSPEC0	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.9 TRCIDR0, 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.13.9 TRCIDR0, ID Register 0 on page 428 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1E0

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-142: ext_trcidr0 bit assignments

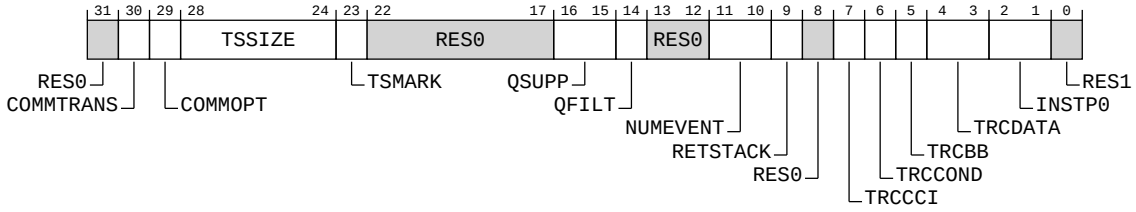


Table B-292: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31]	**RES0**	Reserved	**RES0**
[30]	COMMTRANS	Transaction Start element behavior. **0b0** Transaction Start elements are P0 elements.	x
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **0b1** Commit mode 1. **When ext-TRCIDR8.MAXSPEC == 0x0** Access to this field is: **RAO/WI** **Otherwise** Access to this field is: RO	x

Bits	Name	Description	Reset
[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.	5{x}
[23]	TSMARK	Indicates whether Timestamp Marker elements are generated. **0b1** Timestamp Marker elements are generated.	x
[22:17]	**RES0**	Reserved	**RES0**
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **0b00** Q element support is not implemented.	xx
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	x
[13:12]	**RES0**	Reserved	**RES0**
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **0b11** The trace unit supports 4 ETEEvents.	xx
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **0b1** Return stack implemented.	x
[8]	**RES0**	Reserved	**RES0**
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **0b1** Cycle counting implemented.	x
[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.	x
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. **0b1** Branch broadcasting implemented.	x
[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.	xx
[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.	xx
[0]	**RES1**	Reserved	**RES1**

Accessibility

Component	Offset	Instance	Range
ETE	0x1E0	TRCIDR0	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.10 TRCIDR1, 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.13.10 TRCIDR1, ID Register 1 on page 431 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1E4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-143: ext_trcidr1 bit assignments

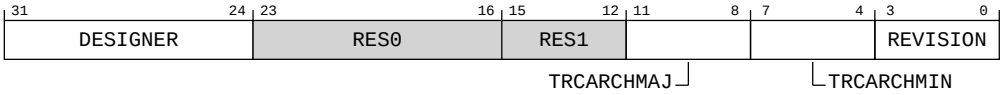


Table B-294: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:24]	DESIGNER	Indicates which company designed the trace unit. The permitted values of this field are the same as AArch64-MIDR_EL1.Implementer. **0b01000001** Arm Limited	8{x}
[23:16]	**RES0**	Reserved	**RES0**
[15:12]	**RES1**	Reserved	**RES1**
[11:8]	TRCARCHMAJ	Major architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	xxxx
[7:4]	TRCARCHMIN	Minor architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	xxxx
[3:0]	REVISION	Arm deprecates any use of this field. **0b0000** r0p2	xxxx

Accessibility

Component	Offset	Instance	Range
ETE	0x1E4	TRCIDR1	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.11 TRCIDR2, 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.13.11 TRCIDR2, ID Register 2 on page 433 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1E8

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-144: ext_trcidr2 bit assignments

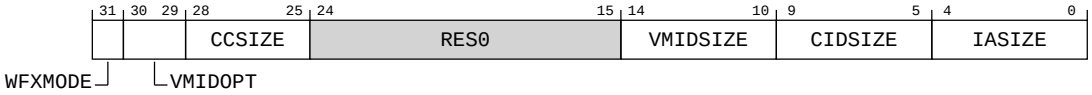


Table B-296: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31]	WFXMODE	Indicates whether WFI and WFE instructions are classified as P0 instructions: **0b1** WFI and WFE instructions are classified as P0 instructions.	x
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. **0b10** Virtual context identifier selection not supported. ext-TRCCONFIGR.VMIDOPT is **RES1**.	xx
[28:25]	CCSIZE	Indicates the size of the cycle counter. **0b0000** The cycle counter is 12 bits in length.	xxxx
[24:15]	**RES0**	Reserved	**RES0**
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. **0b00100** 32-bit Virtual context identifier size.	5{x}

Bits	Name	Description	Reset
[9:5]	CIDSIZE	Indicates the Context identifier size. **0b00100** 32-bit Context identifier size.	5{x}
[4:0]	IASIZE	Virtual instruction address size. **0b01000** Maximum of 64-bit instruction address size.	5{x}

Accessibility

Component	Offset	Instance	Range
ETE	0x1E8	TRCIDR2	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.12 TRCIDR3, 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.13.12 TRCIDR3, ID Register 3 on page 435 bits [31:0].

Attributes

Width
32

Component
ETE

Register offset
0x1EC

Access type
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-145: ext_trcidr3 bit assignments

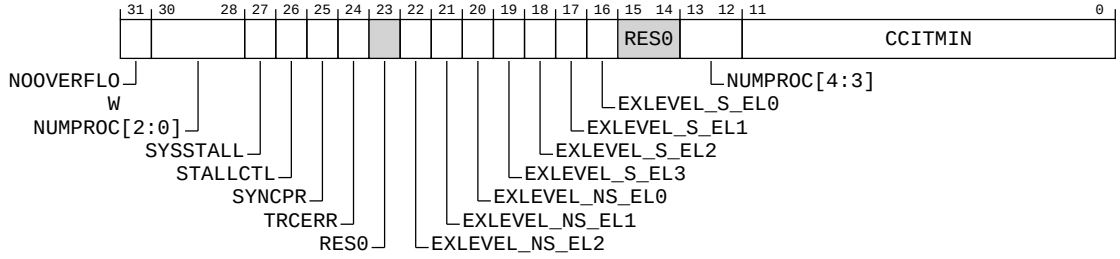


Table B-298: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	x
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **0b1** Stalling of the PE is permitted.	x
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **0b1** Stalling of the PE is implemented.	x
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0b0** ext-TRCSYNCPR is read-write so software can change the synchronization period.	x
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **0b1** Forced tracing of System Error exceptions is implemented.	x
[23]	**RES0**	Reserved	**RES0**
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 is implemented. **0b1** Non-secure EL2 is implemented.	x
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 is implemented. **0b1** Non-secure EL1 is implemented.	x

Bits	Name	Description	Reset
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 is implemented. **0b1** Non-secure EL0 is implemented.	x
[19]	EXLEVEL_S_EL3	Indicates if EL3 is implemented. **0b1** EL3 is implemented.	x
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 is implemented. **0b1** Secure EL2 is implemented.	x
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 is implemented. **0b1** Secure EL1 is implemented.	x
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 is implemented. **0b1** Secure EL0 is implemented.	x
[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.	5{x}
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in ext-TRCCCCTLR.THRESHOLD. If ext-TRCIDR0.TRCCCI == 1 then the minimum value of this field is 0x001. If ext-TRCIDR0.TRCCCI == 0 then this field is zero. **0b000000000100**	12{x}

Accessibility

Component	Offset	Instance	Range
ETE	0x1EC	TRCIDR3	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.13 TRCIDR4, 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.13.13 TRCIDR4, ID Register 4 on page 438 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1F0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-146: ext_trcidr4 bit assignments

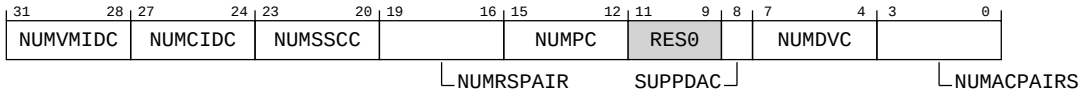


Table B-300: TRCIDR4 bit descriptions

Bits	Name	Description	Reset
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Virtual Context Identifier Comparator.	xxxx
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Context Identifier Comparator.	xxxx

Bits	Name	Description	Reset
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. **0b0001** The implementation has one Single-shot Comparator Control.	xxxx
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0b0111** The implementation has eight resource selector pairs.	xxxx
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0b0000** No PE Comparator Inputs are available.	xxxx
[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.	x
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0000** No data value comparators implemented.	xxxx
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **0b0100** The implementation has four Address Comparator pairs.	xxxx

Accessibility

Component	Offset	Instance	Range
ETE	0x1F0	TRCIDR4	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.14 TRCIDR5, 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.13.14 TRCIDR5, ID Register 5 on page 440 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1F4

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-147: ext_trcidr5 bit assignments

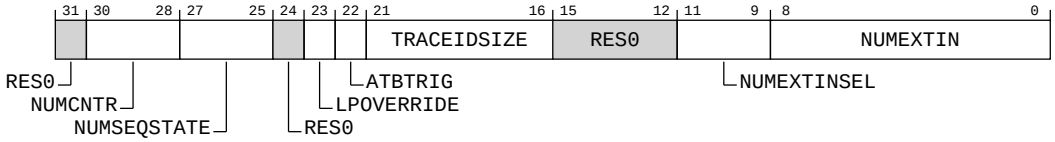


Table B-302: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[31]	**RES0**	Reserved	**RES0**
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. **0b010** Two Counters implemented.	xxx
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **0b100** Four Sequencer states are implemented.	xxx
[24]	**RES0**	Reserved	**RES0**
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **0b1** The trace unit supports Low-power Override Mode.	x

Bits	Name	Description	Reset
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **0b1** The implementation supports ATB triggers.	x
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **0b000111** The implementation supports a 7-bit trace ID.	6{x}
[15:12]	**RES0**	Reserved	**RES0**
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. **0b100** 4 External Input Selector resources are available.	xxx
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **0b111111111** Unified PMU event selection.	9{x}

Accessibility

Component	Offset	Instance	Range
ETE	0x1F4	TRCIDR5	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.15 TRCIDR6, 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 A.13.15 TRCIDR6, ID Register 6 on page 442 bits [31:0].

Attributes

Width
32

Component
ETE

Register offset
0x1F8

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-148: ext_trcidr6 bit assignments

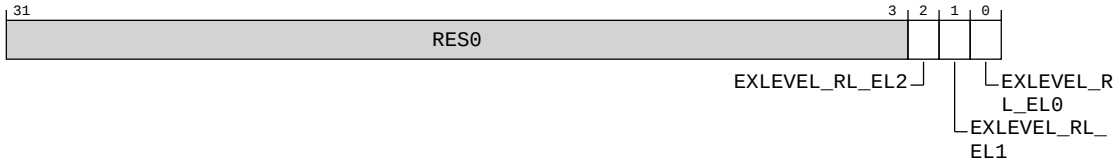


Table B-304: TRCIDR6 bit descriptions

Bits	Name	Description	Reset
[31:3]	**RES0**	Reserved	**RES0**
[2]	EXLEVEL_RL_EL2	Indicates if Realm EL2 is implemented. **0b0** Realm EL2 is not implemented.	x
[1]	EXLEVEL_RL_EL1	Indicates if Realm EL1 is implemented. **0b0** Realm EL1 is not implemented.	x
[0]	EXLEVEL_RL_EL0	Indicates if Realm EL0 is implemented. **0b0** Realm EL0 is not implemented.	x

Accessibility

Component	Offset	Instance	Range
ETE	0x1F8	TRCIDR6	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.16 TRCIDR7, 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 A.13.16 TRCIDR7, ID Register 7 on page 444 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0x1FC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-149: ext_trcidr7 bit assignments

Table B-306: 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() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.17 TRCITATBIDR, Trace Intergration ATB Identification Register

Controls the ATIDM[6:0] signals when TRCITCTRL.IME is set.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xEE4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-150: ext_trcitatbidr bit assignments

Table B-308: TRCITATBIDR bit descriptions

Bits	Name	Description	Reset
[31:7]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[6:0]	TRACEID	Trace ID field. Sets the trace ID value for instruction trace. The width of the field is indicated by the value of ext-TRCIDR5.TRACEIDSIZE. Unimplemented bits are **RES0**. If an implementation supports AMBA ATB, then: • The width of the field is 7 bits. • Writing a reserved trace ID value does not affect behavior of the trace unit but it might cause **UNPREDICTABLE** behavior of the trace capture infrastructure. See the AMBA ATB Protocol Specification for information about which ATID values are reserved.	7{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	0xEE4	TRCITATBIDR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

WO

B.7.18 TRCITATBDATAR, Trace Integration Test ATB Data Register 0

Controls signal outputs when TRCITCTRL.IME is set.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xEEC

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-151: ext_trcitatbdatar bit assignments

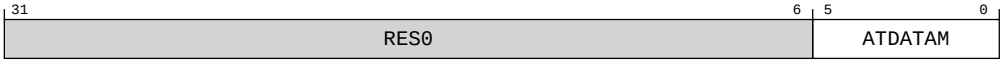


Table B-310: TRCITATBDATAR bit descriptions

Bits	Name	Description	Reset
[31:6]	**RES0**	Reserved	**RES0**
[5:0]	ATDATAM	**When Text("topology detection or integration functionality is implemented")** Drives the ATDATAM **Otherwise** RES0	6{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	0xEEC	TRCITATBDATAR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()
ERROR

Otherwise
WO

B.7.19 TRCITATBINR, Trace Integration ATB In Register

The TRCITIATBINR bit values always correspond to the physical state of the input pins.The TRCITIATBINR reads the state of the input pins.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xEF4

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-152: ext_trcitatbinr bit assignments

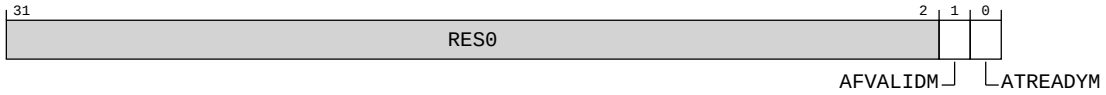


Table B-312: TRCITATBINR bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	AFVALIDM	Returns the value of the AFVALIDMI input pin. **0b0** Input pin is LOW, the corresponding register bit is 0. **0b1** Input pin is HIGH, the corresponding register bit is 1.	x
[0]	ATREADYM	Returns the value of the ATREADYMI input pin. **0b0** Input pin is LOW, the corresponding register bit is 0. **0b1** Input pin is HIGH, the corresponding register bit is 1.	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	0xEF4	TRCITATBINR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.20 TRCITATBOUTR, Trace Integration ATB Out Register

The TRCITIATBOUTR sets the state of the output pins.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xEFC

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-153: ext_trcitatboutr bit assignments

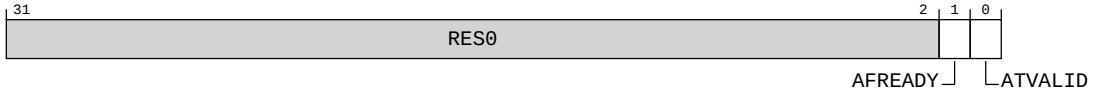


Table B-314: TRCITATBOUTR bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	AFREADY	Drives the AFREADYMI output pin.	x
[0]	ATVALID	Drives the ATVALIDMI output pin.	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	0xEFC	TRCITATBOUTR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

WO

B.7.21 TRCITCTRL, Integration Mode Control Register

A component can use TRCITCTRL to dynamically switch between functional mode and integration mode. In integration mode, topology detection is enabled. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xF00

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-154: ext_trcitctrl bit assignments

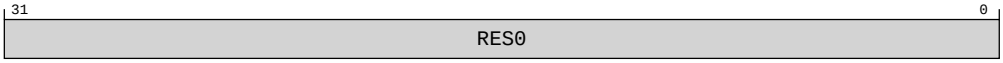


Table B-316: TRCITCTRL bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

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() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.22 TRCCLAIMSET, Claim Tag Set Register

In conjunction with ext-TRCCLAIMCLR, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information, see the CoreSight Architecture Specification.

Configurations

The number of claim tag bits implemented is IMPLEMENTATION DEFINED. Arm recommends that implementations support a minimum of four claim tag bits, that is, SET[3:0] reads as 0b1111.

External register TRCCLAIMSET bits [31:0] are architecturally mapped to AArch64 System register A.13.18 TRCCLAIMSET, Claim Tag Set Register on page 448 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0xFA0

Access type

RAOW1S

Reset value

0000 0000 0000 0000 0000 0000 0000 1111

Bit descriptions

Figure B-155: ext_trcclaimset bit assignments

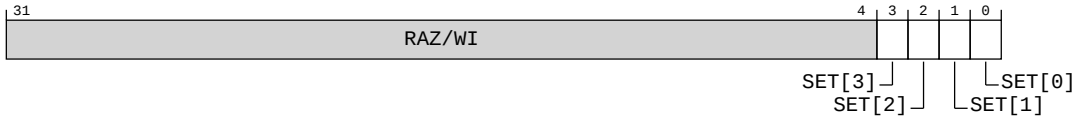


Table B-318: TRCCLAIMSET bit descriptions

Bits	Name	Description	Reset
[31:4]	RAZ/WI	Reserved	RAZ/WI
[3]	SET[3]	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.	0b1
[2]	SET[2]	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.	0b1

Bits	Name	Description	Reset
[1]	SET[1]	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.	0b1
[0]	SET[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.	0b1

Accessibility

Component	Offset	Instance	Range
ETE	0xFA0	TRCCLAIMSET	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RW

B.7.23 TRCCLAIMCLR, Claim Tag Clear Register

In conjunction with ext-TRCCLAIMSET, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information, see the CoreSight Architecture Specification.

Configurations

External register TRCCLAIMCLR bits [31:0] are architecturally mapped to AArch64 System register A.13.19 TRCCLAIMCLR, Claim Tag Clear Register on page 451 bits [31:0].

Attributes

Width
32

Component

ETE

Register offset

0xFA4

Access type

RW1C

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-156: ext_trcclaimclr bit assignments

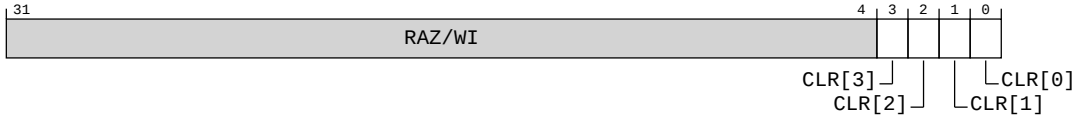


Table B-320: TRCCLAIMCLR bit descriptions

Bits	Name	Description	Reset
[31:4]	**RAZ/WI**	Reserved	**RAZ/WI**
[3]	CLR[3]	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.	0b0
[2]	CLR[2]	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.	0b0

Bits	Name	Description	Reset
[1]	CLR[1]	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.	0b0
[0]	CLR[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.	0b0

Accessibility

Component	Offset	Instance	Range
ETE	0xFA4	TRCCLAIMCLR	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RW

B.7.24 TRCDEVARCH, 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 A.13.20 TRCDEVARCH, Device Architecture Register on page 455 bits [31:0].

Attributes

Width
32

Component

ETE

Register offset

0xFBC

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-157: ext_trcdevarch bit assignments

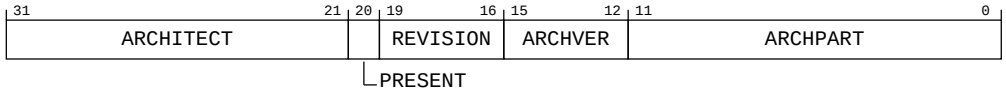


Table B-322: TRCDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code. **0b01000111011** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited. Other values are defined by the JEDEC JEP106 standard. This field reads as 0x23B.	11{x}
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. **0b1** Device Architecture information present.	x
[19:16]	REVISION	Revision. Defines the architecture revision of the component. **0b0001** ETEv1.1, FEAT_ETEv1p1.	xxxx

Bits	Name	Description	Reset
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **0b0101** ETEv1. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12]. This field reads as `0x5`.	`xxxx`
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **0b101000010011** Arm PE trace architecture. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHPART is ARCHID[11:0]. This field reads as `0xA13`.	`12{x}`

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.7.25 TRCDEVID2, Device Configuration Register 2

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFC0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-158: ext_trcdevid2 bit assignments

Table B-324: 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.7.26 TRCDEVID1, Device Configuration Register 1

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFC4

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-159: ext_trcdevid1 bit assignments

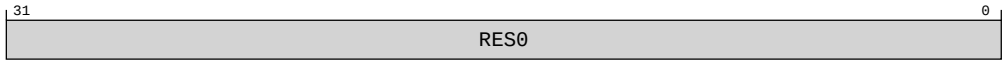


Table B-326: TRCDEVID1 bit descriptions

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.7.27 TRCDEVID, 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 A.13.17 TRCDEVID, Device Configuration Register on page 446 bits [31:0].

Attributes

Width

32

Component

ETE

Register offset

0xFC8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-160: ext_trcdevid bit assignments

Table B-328: 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.7.28 TRCDEVTYPE, Device Type Register

Provides discovery information for the component. If the part number field is not recognized, a debugger can report the information that is provided by TRCDEVTYPE about the component instead.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFCC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-161: ext_trcdevtype bit assignments

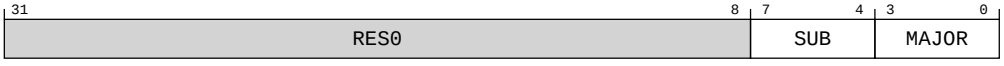


Table B-330: TRCDEVTYPE bit descriptions

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. This field reads as `0x1`.	xxxx
[3:0]	MAJOR	Component major type. **0b0011** Trace source. Other values are defined by the CoreSight Architecture. This field reads as `0x3`.	xxxx

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.7.29 TRCPIDR4, Peripheral Identification Register 4

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-162: ext_trcpidr4 bit assignments

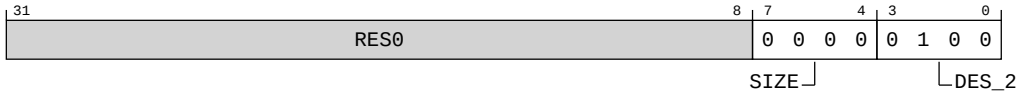


Table B-332: TRCPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	Size of the component. The distance from the start of the address space used by this component to the end of the component identification registers. A value of `0b0000` means one of the following is true: • The component uses a single 4KB block. • The component uses an **IMPLEMENTATION DEFINED** number of 4KB blocks. Any other value means the component occupies $2^{TRCPIDR4.SIZE}$ 4KB blocks. **`0b0000`** Using this field to indicate the size of the component is deprecated. This field might not correctly indicate the size of the component. Arm recommends that software determine the size of the component from the Unique Component Identifier fields, and other **IMPLEMENTATION DEFINED** registers in the component.	`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

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.7.30 TRCPIDR5, Peripheral Identification Register 5

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-163: ext_trcpidr5 bit assignments

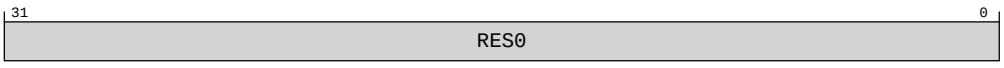


Table B-334: TRCPIDR5 bit descriptions

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.7.31 TRCPIDR6, Peripheral Identification Register 6

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-164: ext_trcpidr6 bit assignments

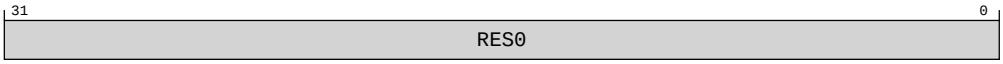


Table B-336: TRCPIDR6 bit descriptions

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.7.32 TRCPIDR7, Peripheral Identification Register 7

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFDC

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-165: ext_trcpidr7 bit assignments

Table B-338: 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.7.33 TRCPIDR0, Peripheral Identification Register 0

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1000 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-166: ext_trcpidr0 bit assignments

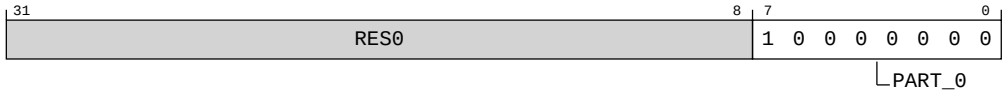


Table B-340: TRCPIDR0 bit descriptions

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 ext-TRCPIDR1.PART_1 and TRCPIDR0.PART_0. **0b10000000** Cortex-A520	0x80

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.7.34 TRCPIDR1, Peripheral Identification Register 1

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-167: ext_trcpidr1 bit assignments

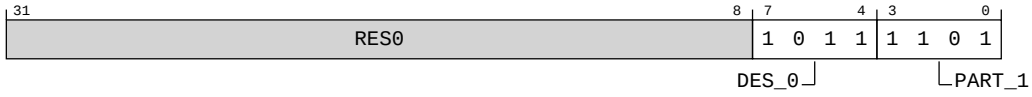


Table B-342: TRCPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7:4]	DES_0	Designer, JEP106 identification code, bits [3:0]. TRCPIDR1.DES_0 and ext-TRCPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. **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 ext-TRCPIDR0.PART_0. **0b1101** Cortex-A520	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.7.35 TRCPIDR2, Peripheral Identification Register 2

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFE8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-168: ext_trcpidr2 bit assignments

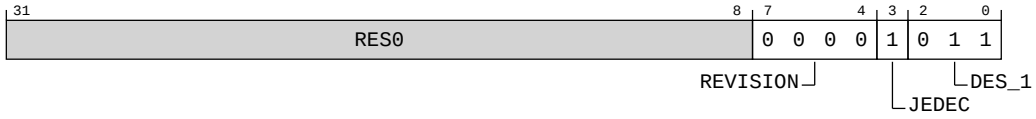


Table B-344: TRCPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Component major revision. TRCPIDR2.REVISION and ext-TRCPIDR3.REVAND together form the revision number of the component, with TRCPIDR2.REVISION being the most significant part and ext-TRCPIDR3.REVAND the least significant part. When a component is changed, TRCPIDR2.REVISION or ext-TRCPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. ext-TRCPIDR3.REVAND should be set to `0b0000` when TRCPIDR2.REVISION is increased. **`0b0000`** r0p2	`0b0000`
[3]	JEDEC	JEDEC-assigned JEP106 implementer code is used. **`0b1`**	`0b1`
[2:0]	DES_1	Designer, JEP106 identification code, bits [6:4]. ext-TRCPIDR1.DES_0 and TRCPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. **`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.7.36 TRCPIDR3, Peripheral Identification Register 3

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFEC

Access type
See bit descriptions

Reset value
xxxx xxxx xxxx xxxx xxxx xxxx 0010 0000

Note
Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-169: ext_trcpidr3 bit assignments

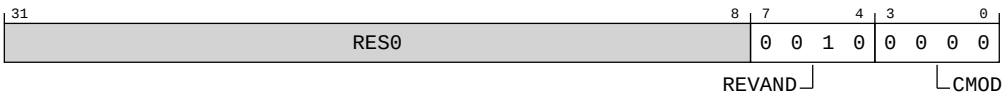


Table B-346: TRCPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	REVAND	Component minor revision. ext-TRCPIDR2.REVISION and TRCPIDR3.REVAND together form the revision number of the component, with ext-TRCPIDR2.REVISION being the most significant part and TRCPIDR3.REVAND the least significant part. When a component is changed, ext-TRCPIDR2.REVISION or TRCPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. TRCPIDR3.REVAND should be set to `0b0000` when ext-TRCPIDR2.REVISION is increased. **`0b0010`** r0p2	`0b0010`
[3:0]	CMOD	Customer Modified. Indicates the component has been modified. A value of `0b0000` means the component is not modified from the original design. Any other value means the component has been modified in an **IMPLEMENTATION DEFINED** way. **`0b0000`**	`0b0000`

Accessibility

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.7.37 TRCCIDR0, Component Identification Register 0

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFF0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-170: ext_trccidr0 bit assignments

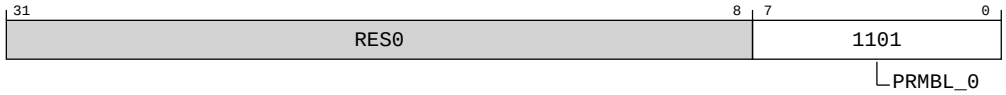


Table B-348: TRCCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Component identification preamble, segment 0. **0b00001101**	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.7.38 TRCCIDR1, Component Identification Register 1

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-171: ext_trccidr1 bit assignments

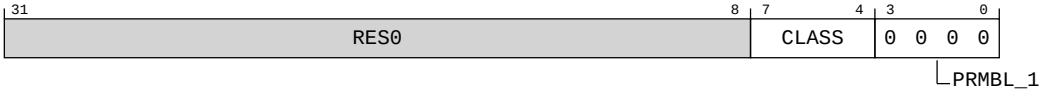


Table B-350: TRCCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight peripheral. Other values are defined by the CoreSight Architecture. This field reads as 0x9.	xxxx
[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.7.39 TRCCIDR2, Component Identification Register 2

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-172: ext_trccidr2 bit assignments

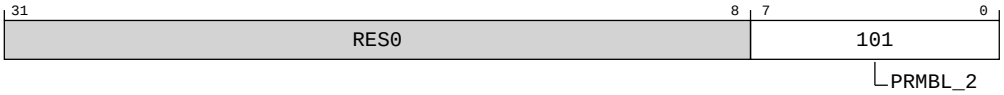


Table B-352: TRCCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Component identification preamble, segment 2. **0b00000101**	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.7.40 TRCCIDR3, Component Identification Register 3

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFFC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-173: ext_trccidr3 bit assignments

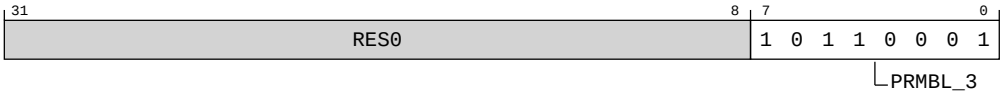


Table B-354: TRCCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Component identification preamble, segment 3. **0b10110001**	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.8 External ROM table registers summary

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

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

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-356: ROM table registers summary

Offset	Name	Reset	Width	Description
0x0	ROMENTRY0	–	32-bit	Class 0x9 ROM Table Entries
0x4	ROMENTRY1	–	32-bit	Class 0x9 ROM Table Entries
0x8	ROMENTRY2	–	32-bit	Class 0x9 ROM Table Entries
0xC	ROMENTRY3	–	32-bit	Class 0x9 ROM Table Entries
0x10	ROMENTRY4	–	32-bit	Class 0x9 ROM Table Entries
0x14	ROMENTRY5	–	32-bit	Class 0x9 ROM Table Entries
0x18	ROMENTRY6	–	32-bit	Class 0x9 ROM Table Entries
0x1C	ROMENTRY7	–	32-bit	Class 0x9 ROM Table Entries
0xF00	ITCTRL	–	32-bit	Integration Mode Control Register
0xFA0	CLAIMSET	–	32-bit	Claim Tag Set Register
0xFA4	CLAIMCLR	–	32-bit	Claim Tag Clear Register
0xFA8	DEVAFF0	–	32-bit	Device Affinity Register 0
0xFAC	DEVAFF1	–	32-bit	Device Affinity Register 1
0xFB0	LAR	–	32-bit	Software Lock Access Register
0xFB4	LSR	–	32-bit	Software Lock Status Register
0xFB8	AUTHSTATUS	–	32-bit	Authentication Status Register
0xFBC	DEVARCH	–	32-bit	Device Architecture Register
0xFC0	DEVID2	–	32-bit	Device Configuration Register 2
0xFC4	DEVID1	–	32-bit	Device Configuration Register 1
0xFC8	DEVID	–	32-bit	Device Configuration Register
0xFCC	DEVTYPE	–	32-bit	Device Type Register
0xFD0	PIDR4	–	32-bit	Peripheral Identification Register 4
0xFD4	PIDR5	–	32-bit	Peripheral Identification Register 5
0xFD8	PIDR6	–	32-bit	Peripheral Identification Register 6
0xFDC	PIDR7	–	32-bit	Peripheral Identification Register 7
0xFE0	PIDR0	–	32-bit	Peripheral Identification Register 0
0xFE4	PIDR1	–	32-bit	Peripheral Identification Register 1
0xFE8	PIDR2	–	32-bit	Peripheral Identification Register 2
0xFEC	PIDR3	–	32-bit	Peripheral Identification Register 3
0xFF0	CIDR0	–	32-bit	Component Identification Register 0
0xFF4	CIDR1	–	32-bit	Component Identification Register 1
0xFF8	CIDR2	–	32-bit	Component Identification Register 2
0xFFC	CIDR3	–	32-bit	Component Identification Register 3

B.8.1 ROMENTRY0, Class 0x9 ROM Table Entries

A Class 0x9 ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component <n>, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class 0x9 ROM Table:

- The first entry, entry 0, has offset 0x000.
- ext-ROMENTRY<n> has the offset 0x000 + <n>×4, where 0 ≤ <n> ≤ 511.
- If the number of components, N, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:
  - ROM Table entries representing components have offsets from 0x000 to (N-1)×4.
  - The ext-ROMENTRY<n> at offset N×4, which has a PRESENT field with the value 0b00, indicates the end of the ROM Table.
- If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from 0x000 to 0x7FC. If a ROM Table entry is present at offset 0x7FC, its PRESENT field must have a value of either 0b00 or 0b11, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value 0b11.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

If ext-DEVID.FORMAT has the value 0x0, ext-ROMENTRY<n> are 512 32-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0x0

Access type

Read

R

Write

RESERVED

Reset value

0000 0000 0000 0001 0000 xxxx xxxx x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-174: ext_romentry0 bit assignments

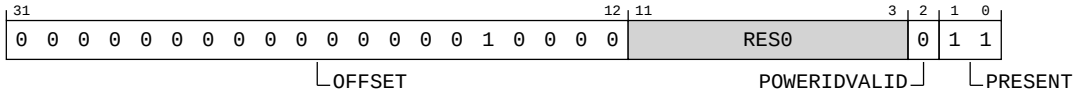


Table B-357: ROMENTRY0 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **0b00000000000000010000** Core 0 Debug	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

A ROM Table does not have to use power domain IDs. If none of the ROM Table entries provides a power domain ID, all the components that pointed to by the ROM Table are in the same power domain as the ROM Table.

Component	Offset
ROM table	0x0

This interface is accessible as follows:

RO

B.8.2 ROMENTRY1, Class 0x9 ROM Table Entries

A Class 0x9 ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component <n>, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class 0x9 ROM Table:

- The first entry, entry 0, has offset 0x000.
- ext-ROMENTRY<n> has the offset 0x000 + <n>×4, where 0 ≤ <n> ≤ 511.
- If the number of components, N, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:
  - ROM Table entries representing components have offsets from 0x000 to (N-1)×4.
  - The ext-ROMENTRY<n> at offset N×4, which has a PRESENT field with the value 0b00, indicates the end of the ROM Table.
- If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from 0x000 to 0x7FC. If a ROM Table entry is present at offset 0x7FC, its PRESENT field must have a value of either 0b00 or 0b11, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value 0b11.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

If ext-DEVID.FORMAT has the value 0x0, ext-ROMENTRY<n> are 512 32-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0x4

Access type

Read

R

Write

RESERVED

Reset value

0000 0000 0000 0010 0000 xxxx xxxx x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-175: ext_romentry1 bit assignments

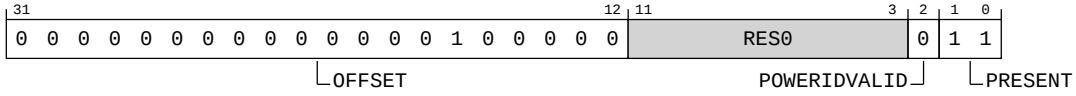


Table B-359: ROMENTRY1 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **0b00000000000000100000** Core 0 PMU	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

A ROM Table does not have to use power domain IDs. If none of the ROM Table entries provides a power domain ID, all the components that pointed to by the ROM Table are in the same power domain as the ROM Table.

Component	Offset
ROM table	0x4

This interface is accessible as follows:

RO

B.8.3 ROMENTRY2, Class 0x9 ROM Table Entries

A Class 0x9 ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component <n>, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class 0x9 ROM Table:

- The first entry, entry 0, has offset 0x000.
- ext-ROMENTRY<n> has the offset 0x000 + <n>×4, where 0 ≤ <n> ≤ 511.
- If the number of components, N, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:
  - ROM Table entries representing components have offsets from 0x000 to (N-1)×4.
  - The ext-ROMENTRY<n> at offset N×4, which has a PRESENT field with the value 0b00, indicates the end of the ROM Table.
- If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from 0x000 to 0x7FC. If a ROM Table entry is present at offset 0x7FC, its PRESENT field must have a value of either 0b00 or 0b11, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value 0b11.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

If ext-DEVID.FORMAT has the value 0x0, ext-ROMENTRY<n> are 512 32-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0x8

Access type

Read

R

Write

RESERVED

Reset value

0000 0000 0000 0011 0000 xxxx xxxx x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-176: ext_romentry2 bit assignments

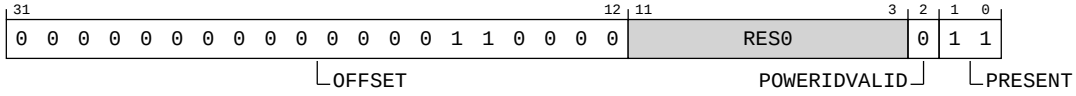


Table B-361: ROMENTRY2 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **0b00000000000000110000** Core 0 trace unit	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

A ROM Table does not have to use power domain IDs. If none of the ROM Table entries provides a power domain ID, all the components that pointed to by the ROM Table are in the same power domain as the ROM Table.

Component	Offset
ROM table	0x8

This interface is accessible as follows:

RO

B.8.4 ROMENTRY3, Class 0x9 ROM Table Entries

A Class 0x9 ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component <n>, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class 0x9 ROM Table:

- The first entry, entry 0, has offset 0x000.
- ext-ROMENTRY<n> has the offset 0x000 + <n>×4, where 0 ≤ <n> ≤ 511.
- If the number of components, N, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:
  - ROM Table entries representing components have offsets from 0x000 to (N-1)×4.
  - The ext-ROMENTRY<n> at offset N×4, which has a PRESENT field with the value 0b00, indicates the end of the ROM Table.
- If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from 0x000 to 0x7FC. If a ROM Table entry is present at offset 0x7FC, its PRESENT field must have a value of either 0b00 or 0b11, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value 0b11.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

If ext-DEVID.FORMAT has the value 0x0, ext-ROMENTRY<n> are 512 32-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0xC

Access type

Read

R

Write

RESERVED

Reset value

0000 0000 0000 0100 0000 xxxx xxxx x0xx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-177: ext_romentry3 bit assignments

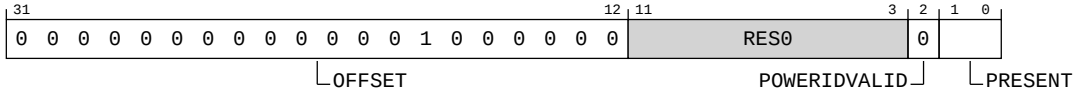


Table B-363: ROMENTRY3 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **0b00000000000001000000** ELA	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. **0b10** The ROM entry is not present. This value is reported when the complex is configured without the ELA. **0b11** The ROM entry is present. This value is reported when the complex is configured with the ELA.	The reset values can be the following: 0b10, 0b11, respective to the value.

Accessibility

A ROM Table does not have to use power domain IDs. If none of the ROM Table entries provides a power domain ID, all the components that pointed to by the ROM Table are in the same power domain as the ROM Table.

Component	Offset
ROM table	0xC

This interface is accessible as follows:

RO

B.8.5 ROMENTRY4, Class 0x9 ROM Table Entries

A Class 0x9 ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component <n>, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class 0x9 ROM Table:

- The first entry, entry 0, has offset 0x000.
- ext-ROMENTRY<n> has the offset 0x000 + <n>×4, where 0 ≤ <n> ≤ 511.
- If the number of components, N, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:
  - ROM Table entries representing components have offsets from 0x000 to (N-1)×4.
  - The ext-ROMENTRY<n> at offset N×4, which has a PRESENT field with the value 0b00, indicates the end of the ROM Table.
- If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from 0x000 to 0x7FC. If a ROM Table entry is present at offset 0x7FC, its PRESENT field must have a value of either 0b00 or 0b11, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value 0b11.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

If ext-DEVID.FORMAT has the value 0x0, ext-ROMENTRY<n> are 512 32-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0x10

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x0xx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-178: ext_romentry4 bit assignments

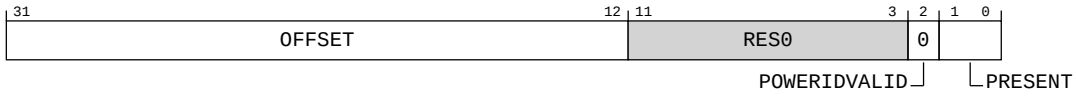


Table B-365: ROMENTRY4 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **0b00000000000000000000** This value is reported when the complex is configured with one core. **0b00000000000010010000** Core 1 Debug. This value is reported when the complex is configured with two cores.	The reset values can be the following: 0b00000000000000000000, 0b00000000000010010000, respective to the value.
[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

Bits	Name	Description	Reset
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **0b00** The ROM entry is not present. This value is reported when the complex is configured with one core. **0b11** The ROM entry is present. This value is reported when the complex is configured with two cores.	The reset values can be the following: 0b00, 0b11, respective to the value.

Accessibility

A ROM Table does not have to use power domain IDs. If none of the ROM Table entries provides a power domain ID, all the components that pointed to by the ROM Table are in the same power domain as the ROM Table.

Component	Offset
ROM table	0x10

This interface is accessible as follows:

RO

B.8.6 ROMENTRY5, Class 0x9 ROM Table Entries

A Class 0x9 ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component <n>, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class 0x9 ROM Table:

- The first entry, entry 0, has offset 0x000.
- ext-ROMENTRY<n> has the offset 0x000 + <n>×4, where 0 ≤ <n> ≤ 511.
- If the number of components, N, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:
  - ROM Table entries representing components have offsets from 0x000 to (N-1)×4.
  - The ext-ROMENTRY<n> at offset N×4, which has a PRESENT field with the value 0b00, indicates the end of the ROM Table.
- If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from 0x000 to 0x7FC. If a ROM Table entry is present at offset 0x7FC, its PRESENT field must have a value of either 0b00 or 0b11, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value 0b11.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

If ext-DEVID.FORMAT has the value 0x0, ext-ROMENTRY<n> are 512 32-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0x14

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x0xx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-179: ext_romentry5 bit assignments

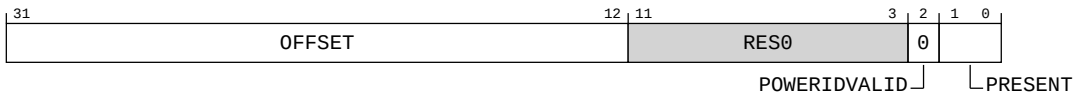


Table B-367: ROMENTRY5 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **0b00000000000000000000** This value is reported when the complex is configured with one core. **0b00000000000010100000** Core 1 PMU. This value is reported when the complex is configured with two cores.	The reset values can be the following: 0b00000000000000000000, 0b00000000000010100000, respective to the value.
[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. **0b00** The ROM entry is not present. This value is reported when the complex is configured with one core. **0b11** The ROM entry is present. This value is reported when the complex is configured with two cores.	The reset values can be the following: 0b00, 0b11, respective to the value.

Accessibility

A ROM Table does not have to use power domain IDs. If none of the ROM Table entries provides a power domain ID, all the components that pointed to by the ROM Table are in the same power domain as the ROM Table.

Component	Offset
ROM table	0x14

This interface is accessible as follows:

RO

B.8.7 ROMENTRY6, Class 0x9 ROM Table Entries

A Class 0x9 ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component <n>, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class 0x9 ROM Table:

- The first entry, entry 0, has offset 0x000.
- ext-ROMENTRY<n> has the offset 0x000 + <n>×4, where 0 ≤ <n> ≤ 511.
- If the number of components, N, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:
  - ROM Table entries representing components have offsets from 0x000 to (N-1)×4.
  - The ext-ROMENTRY<n> at offset N×4, which has a PRESENT field with the value 0b00, indicates the end of the ROM Table.
- If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from 0x000 to 0x7FC. If a ROM Table entry is present at offset 0x7FC, its PRESENT field must have a value of either 0b00 or 0b11, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value 0b11.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

If ext-DEVID.FORMAT has the value 0x0, ext-ROMENTRY<n> are 512 32-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0x18

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x0xx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-180: ext_romentry6 bit assignments

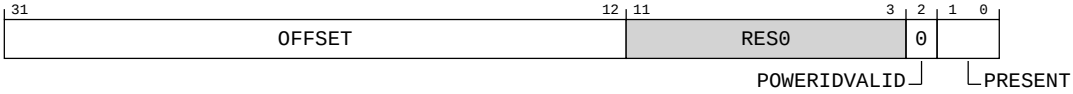


Table B-369: ROMENTRY6 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **0b00000000000000000000** This value is reported when the complex is configured with one core. **0b00000000000010110000** Core 1 trace unit. This value is reported when the complex is configured with two cores.	The reset values can be the following: 0b00000000000000000000, 0b00000000000010110000, respective to the value.
[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. **0b00** The ROM entry is not present. This value is reported when the complex is configured with one core. **0b11** The ROM entry is present. This value is reported when the complex is configured with two cores.	The reset values can be the following: 0b00, 0b11, respective to the value.

Accessibility

A ROM Table does not have to use power domain IDs. If none of the ROM Table entries provides a power domain ID, all the components that pointed to by the ROM Table are in the same power domain as the ROM Table.

Component	Offset
ROM table	0x18

This interface is accessible as follows:

RO

B.8.8 ROMENTRY7, Class 0x9 ROM Table Entries

A Class 0x9 ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component <n>, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class 0x9 ROM Table:

- The first entry, entry 0, has offset 0x000.
- ext-ROMENTRY<n> has the offset 0x000 + <n>×4, where 0 ≤ <n> ≤ 511.
- If the number of components, N, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:
  - ROM Table entries representing components have offsets from 0x000 to (N-1)×4.
  - The ext-ROMENTRY<n> at offset N×4, which has a PRESENT field with the value 0b00, indicates the end of the ROM Table.
- If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from 0x000 to 0x7FC. If a ROM Table entry is present at offset 0x7FC, its PRESENT field must have a value of either 0b00 or 0b11, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value 0b11.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

If ext-DEVID.FORMAT has the value 0x0, ext-ROMENTRY<n> are 512 32-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0x1C

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-181: ext_romentry7 bit assignments

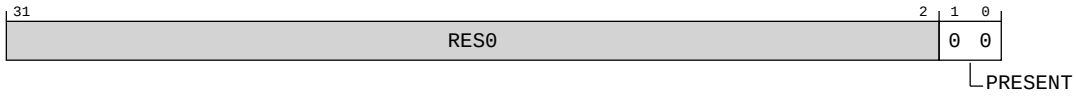


Table B-371: ROMENTRY7 bit descriptions

Bits	Name	Description	Reset
[31:2]	**RES0**	Reserved	**RES0**
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **0b00** The ROM entry is not present, and this ext-ROMENTRY<n> is the final entry in the ROM Table. If PRESENT has this value, all other fields in this ext-ROMENTRY<n> must be zero.	0b00

Accessibility

A ROM Table does not have to use power domain IDs. If none of the ROM Table entries provides a power domain ID, all the components that pointed to by the ROM Table are in the same power domain as the ROM Table.

Component	Offset
ROM table	0x1C

This interface is accessible as follows:

RO

B.8.9 DEVARCH, Device Architecture Register

Identifies the architect and architecture of a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFBC

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-182: ext_devarch bit assignments

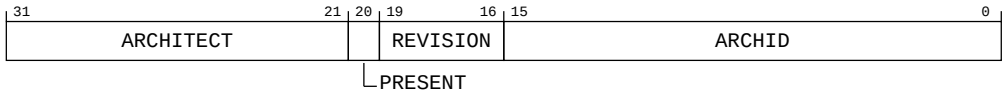


Table B-373: DEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. **0b01000111011** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	11{x}
[20]	PRESENT	Present. **0b1** DEVARCH information present.	x

Bits	Name	Description	Reset
[19:16]	REVISION	Revision. **0b0000** Revision 0.	xxxx
[15:0]	ARCHID	Architecture ID. **0b0000101011110111** ROM Table v0. The debug tool must inspect ext-DEVTYPE and ext-DEVID to determine further information about the ROM Table.	16{x}

Accessibility

Component	Offset
ROM table	0xFBC

This interface is accessible as follows:

RO

B.8.10 DEVID2, Device Configuration Register 2

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFC0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-183: ext_devid2 bit assignments

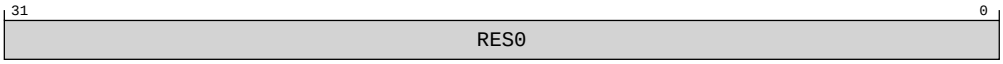


Table B-375: DEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset
ROM table	0xFC0

This interface is accessible as follows:

RO

B.8.11 DEVID1, Device Configuration Register 1

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFC4

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-184: ext_devid1 bit assignments

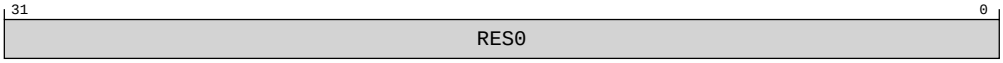


Table B-377: DEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset
ROM table	0xFC4

This interface is accessible as follows:

RO

B.8.12 DEVID, Device Configuration Register

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFC8

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-185: ext_devid bit assignments

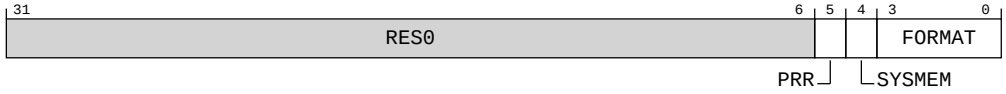


Table B-379: DEVID bit descriptions

Bits	Name	Description	Reset
[31:6]	**RES0**	Reserved	**RES0**
[5]	PRR	Power Request functionality included. **0b0** Power Request functionality not included. If any ROM Table entries contain power domain IDs, a GPR must be present, and pointed to by the ROM Table. The GPR provides functionality to control the power domains. ext-PRIDR0 is not implemented.	x
[4]	SYSMEM	System memory present. Indicates whether system memory is present on the bus that connects to the ROM Table. **0b0** System memory is not present on the bus. This value indicates that the bus is a dedicated debug bus. The ROM Table indicates all the valid addresses in the memory system that the ADI is connected to, and the result of accessing any other address is **UNPREDICTABLE**.	x
[3:0]	FORMAT	ROM format. **0b0000** 32-bit format 0.	xxxx

Accessibility

Component	Offset
ROM table	0xFC8

This interface is accessible as follows:

RO

B.8.13 DEVTYPE, Device Type Register

A debugger can use DEVTYPE to obtain information about a component that has an unrecognized part number.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFCC

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-186: ext_devtype bit assignments

Table B-381: DEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SUB	Sub number **0b0000** Other, undefined.	xxxx
[3:0]	MAJOR	Major number **0b0000** Miscellaneous.	xxxx

Accessibility

Component	Offset
ROM table	0xFCC

This interface is accessible as follows:

RO

B.8.14 PIDR4, Peripheral Identification Register 4

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFD0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-187: ext_pidr4 bit assignments

Table B-383: PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	Size of the component. **RAZ**. Log_2 of the number of 4KB pages from the start of the component to the end of the component ID registers. **0b0000** A ROM Table occupies a single 4KB block of memory.	xxxx
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **0b0100** Arm Limited	xxxx

Accessibility

Component	Offset
ROM table	0xFD0

This interface is accessible as follows:

RO

B.8.15 PIDR5, Peripheral Identification Register 5

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFD4

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-188: ext_pidr5 bit assignments

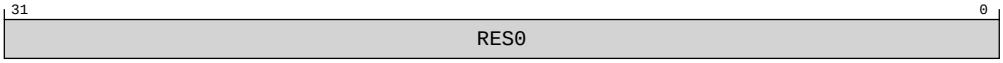


Table B-385: PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset
ROM table	0xFD4

This interface is accessible as follows:

RO

B.8.16 PIDR6, Peripheral Identification Register 6

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFD8

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-189: ext_pidr6 bit assignments

Table B-387: PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset
ROM table	0xFD8

This interface is accessible as follows:

RO

B.8.17 PIDR7, Peripheral Identification Register 7

Provide information to identify a CoreSight componentn.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFDC

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-190: ext_pidr7 bit assignments

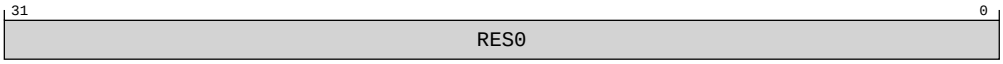


Table B-389: PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset
ROM table	0xFDC

This interface is accessible as follows:

RO

B.8.18 PIDR0, Peripheral Identification Register 0

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFE0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-191: ext_pidr0 bit assignments

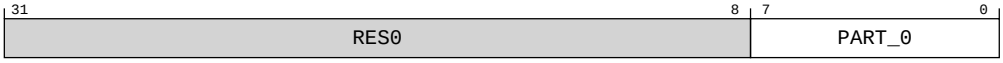


Table B-391: PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b10000000** Cortex-A520	8{x}

Accessibility

Component	Offset
ROM table	0xFE0

This interface is accessible as follows:

RO

B.8.19 PIDR1, Peripheral Identification Register 1

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFE4

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-192: ext_pidr1 bit assignments

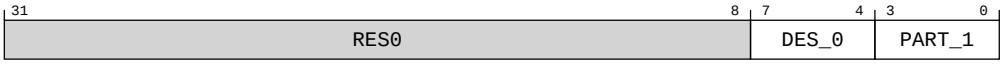


Table B-393: PIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is `0b1011`. **`0b1011`** Arm Limited	xxxx
[3:0]	PART_1	Part number, most significant nibble. **`0b1101`** Cortex-A520	xxxx

Accessibility

Component	Offset
ROM table	0xFE4

This interface is accessible as follows:

RO

B.8.20 PIDR2, Peripheral Identification Register 2

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFE8

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-193: ext_pidr2 bit assignments

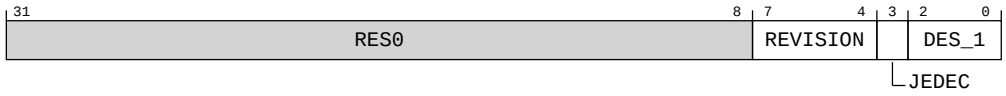


Table B-395: PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0000** r0p2	xxxx
[3]	JEDEC	**RAO**. Indicates a JEP106 identity code is used.	x

Bits	Name	Description	Reset
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is `0b011`. **`0b011`** Arm Limited	`xxx`

Accessibility

Component	Offset
ROM table	0xFE8

This interface is accessible as follows:

RO

B.8.21 PIDR3, Peripheral Identification Register 3

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFEC

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-194: ext_pidr3 bit assignments

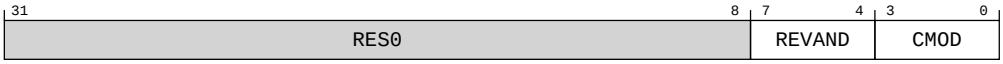


Table B-397: PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Part minor revision. Parts using ext-PIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0010** r0p2	xxxx
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000**	xxxx

Accessibility

Component	Offset
ROM table	0xFEC

This interface is accessible as follows:

RO

B.8.22 CIDR0, Component Identification Register 0

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFF0

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-195: ext_cidr0 bit assignments

Table B-399: CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	CoreSight component identification preamble. **0b00001101** CoreSight component identification preamble.	8{x}

Accessibility

Component	Offset	Instance	Range
ROM table	0xFF0	CIDR0	None

This interface is accessible as follows:

RO

B.8.23 CIDR1, Component Identification Register 1

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFF4

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-196: ext_cidr1 bit assignments

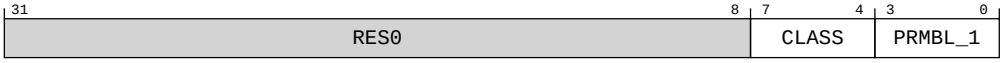


Table B-401: CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	CoreSight component class. **0b1001** CoreSight component.	xxxx
[3:0]	PRMBL_1	CoreSight component identification preamble. **0b0000** CoreSight component identification preamble.	xxxx

Accessibility

Component	Offset
ROM table	0xFF4

This interface is accessible as follows:

RO

B.8.24 CIDR2, Component Identification Register 2

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFF8

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-197: ext_cidr2 bit assignments

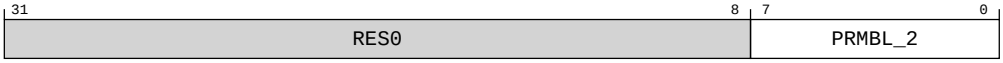


Table B-403: CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	CoreSight component identification preamble. **0b00000101** CoreSight component identification preamble.	8{x}

Accessibility

Component	Offset
ROM table	0xFF8

This interface is accessible as follows:

RO

B.8.25 CIDR3, Component Identification Register 3

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFFC

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-198: ext_cidr3 bit assignments

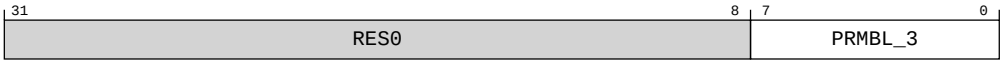


Table B-405: CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	CoreSight component identification preamble. **0b10110001** CoreSight component identification preamble.	8{x}

Accessibility

Component	Offset
ROM table	0xFFC

This interface is accessible as follows:

RO

Appendix C Document revisions

This appendix records the changes between released issues of this document.

C.1 Revisions

Changes between released issues of this book are summarized in tables.

The first table is for the first release. Then, each table compares the new issue of the book with the last released issue of the book. Release numbers match the revision history in Release Information on page 2.

Table C-1: Issue 0000-01

Change	Location
First beta release for r0p0	-

Table C-2: Differences between issue 0000-01 and issue 0000-02

Change	Location
First limited access release for r0p0	-
Various additions and clarifications	2.4 Supported standards and specifications on page 28
Clarifications to External aborts	6.5 Responses on page 66
Added new section	6.7 Page-based hardware attributes on page 68
Added new chapter	12. Utility bus on page 93
Clarifications and additions to Architectural PMU events table	18.1 Performance monitors events on page 113
Clarifications and additions to Arm **IMPLEMENTATION DEFINED** PMU events table	18.1.2 implementation defined performance monitors events on page 132
Added new registers to the Generic system control registers summary table	A.1 AArch64 Generic System Control registers summary on page 163
Added new registers to the Debug registers summary table	A.3 AArch64 Debug registers summary on page 238
Added new registers to the GIC registers summary table	A.6 AArch64 GIC system registers summary on page 327
Added new registers to the Performance Monitors registers summary table	A.7 AArch64 Performance Monitors registers summary on page 357
Added new registers to the RAS registers summary table	A.12 AArch64 RAS registers summary on page 406
Added new registers to the Activity Monitors registers summary table	B.6 External AMU registers summary on page 679
Added new registers to the ETE registers summary table	B.7 External ETE registers summary on page 712

Table C-3: Differences between issue 0000-02 and issue 0001-03

Change	Location
First early access release for r0p1	-
Editorial changes	Throughout the document
Added FEAT_ECBHB to Table 2-2	2.4 Supported standards and specifications on page 28
Added FEAT_Debugv8p4.Debug	2.4 Supported standards and specifications on page 28

Change	Location
Added 'Related information' to power control chapter	5.4 Core power modes on page 50
Updated the section on Core powerup and powerdown sequence	5.7 Cortex-A520 core powerup and powerdown sequence on page 58
Updated the notes	8.1 L1 data cache behavior on page 73
Updated the descriptions of SED parity and SECDED ECC	11.1 Cache protection behavior on page 86
Added new section into the Debug chapter	17.7 CTI register identification values on page 109
Clarified the section and the table title	18.1.1 Common event PMU events on page 113
Updated the Arm **IMPLEMENTATION DEFINED** PMU events table	18.1.2 implementation defined performance monitors events on page 132
Removed the register TRCAUTHSTATUS	19.8 AArch64 Trace unit registers on page 149
Updated the bit assignment figures and the bit description tables for the IMP_CDBGDR0_EL3 register	A.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0 on page 239
Added the following registers: • TRCITATBIDR • TRCITATBDATAR • TRCITATBINR • TRCITATBOUTR	• B.7.17 TRCITATBIDR, Trace Intergration ATB Identification Register on page 739 • B.7.18 TRCITATBDATAR, Trace Integration Test ATB Data Register 0 on page 740 • B.7.19 TRCITATBINR, Trace Integration ATB In Register on page 741 • B.7.20 TRCITATBOUTR, Trace Integration ATB Out Register on page 743

Table C-4: Differences between issue 0001-03 and issue 0001-04

Change	Location
Second early access release for r0p1	-
Editorial changes	Throughout the document
Updated product name	Throughout the document
Updated Common event PMU events table	18.1.1 Common event PMU events on page 113
Updated the Arm **IMPLEMENTATION DEFINED** PMU events table	18.1.2 implementation defined performance monitors events on page 132

Table C-5: Differences between issue 0001-04 and issue 0002-05

Change	Location
First early access release for r0p2	-
Editorial changes	Throughout the document
Clarified the section and the table titles	2.4 Supported standards and specifications on page 28
Updated the section on Warm reset mode	5.4.7 Warm reset mode on page 55
Updated Common event PMU events table	18.1.1 Common event PMU events on page 113
Updated CoreSight component identification table	17.6 CoreSight component identification on page 109
Updated CTI register peripheral ID values table	17.7 CTI register identification values on page 109
Updated the Arm **IMPLEMENTATION DEFINED** PMU events table	18.1.2 implementation defined performance monitors events on page 132
Updated IMP_CDBGDR0_EL3 bit descriptions	A.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0 on page 239
Updated identification registers summary table	A.5 AArch64 Identification registers summary on page 268