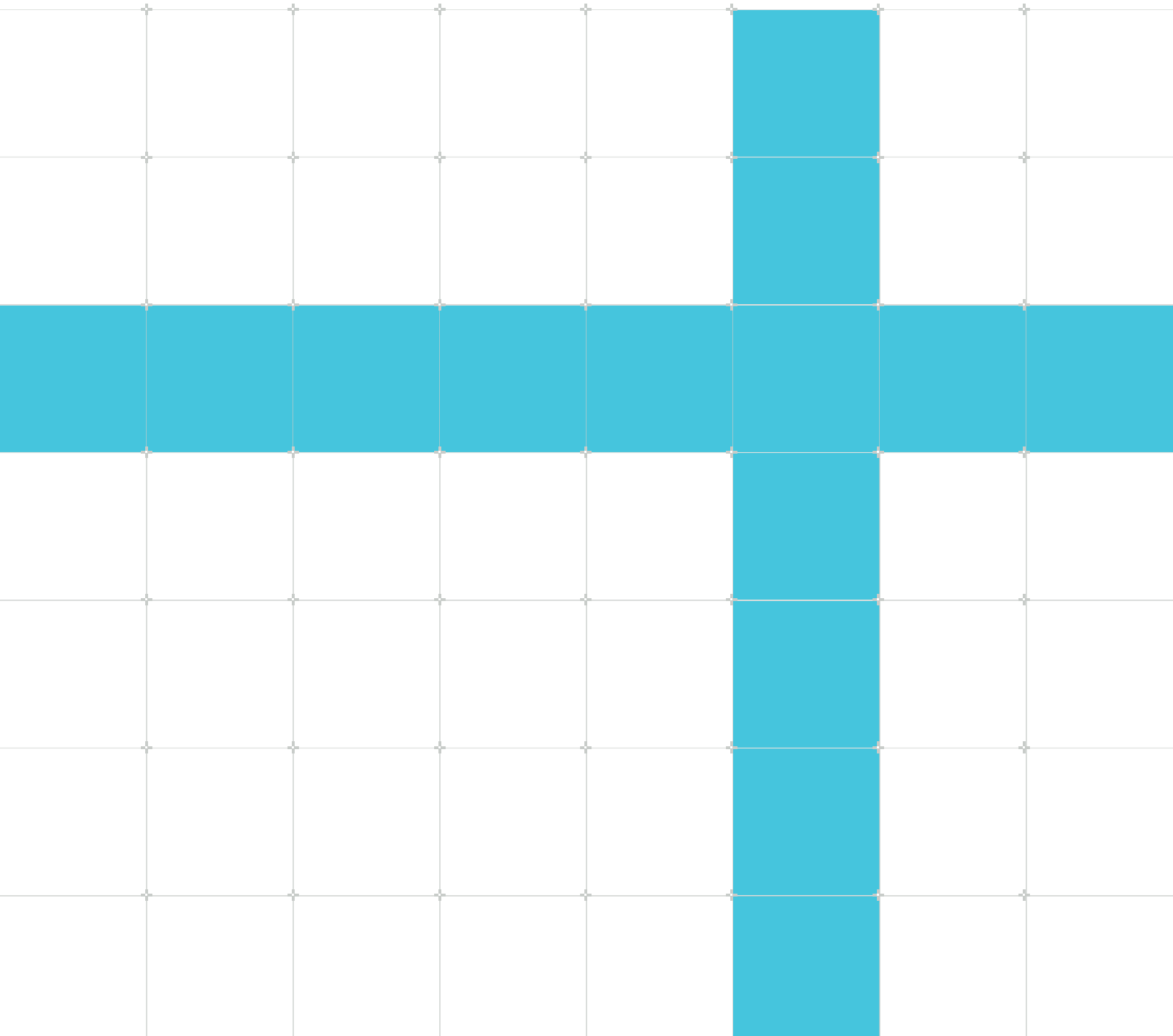
Arm® Cortex®-A715 Core

Revision: r1p2

Technical Reference Manual

Issue 08
101590_0102_08_en



Arm® Cortex®-A715 Core

Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-02	17 July 2020	Confidential	First alpha release for r0p0
0000-03	30 November 2020	Confidential	First beta release for r0p0
0000-04	7 May 2021	Confidential	First limited access release for r0p0
0100-05	30 September 2021	Confidential	First early access release for r1p0
0101-06	25 May 2022	Confidential	First early access release for r1p1
0101-07	22 June 2022	Non-Confidential	Second early access release for r1p1
0102-08	30 September 2022	Non-Confidential	First release for r1p2

Proprietary Notice

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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

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

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.

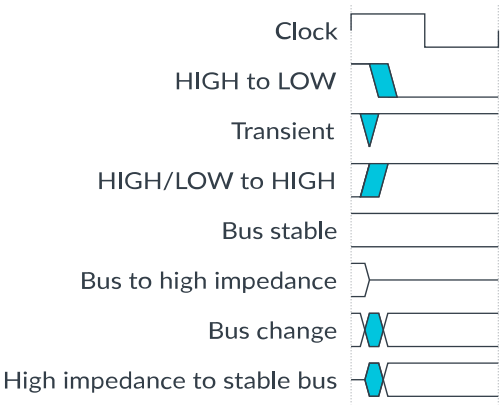
A reminder of something important that relates to the information you are reading.

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{0}` 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® Cortex®-A715 Core Configuration and Integration Manual	101591	Confidential
Arm® Cortex®-A715 Core Cryptographic Extension Technical Reference Manual	101592	Non-Confidential
Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual	101381	Non-Confidential
Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual	101382	Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® 5 CHI Architecture Specification	IHI 0050	Non-Confidential
Arm® Architecture Reference Manual Supplement, The Scalable Vector Extension	DDI 0584	Non-Confidential
Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile	DDI0608	Non-Confidential
Arm® Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM) for Armv8-A	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 Technical Reference Manual	101088	Non-Confidential
Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4	IHI 0069	Non-Confidential
Arm® Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile	DDI 0587	Non-Confidential

Non-Arm resources	Document ID	Organization
-	-	-

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

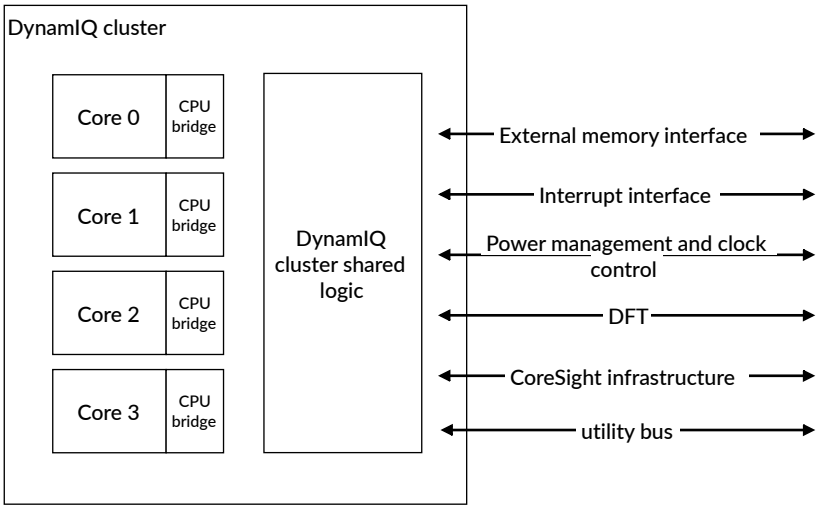
2. The Cortex®-A715 core

The Cortex®-A715 core is a balanced-performance, low-power, and constrained area product that implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.5-A. The Cortex®-A715 core targets large screen compute applications as well as smartphone applications.

The Cortex®-A715 core is implemented inside a DSU-110 DynamIQ™ cluster. It is connected to the *DynamIQ™ Shared Unit-110* (DSU-110) 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®-A715 core is the balanced-performance core.

The following figure shows an example configuration with four Cortex®-A715 cores in a DynamIQ™ cluster.

Figure 2-1: Cortex®-A715 cores example configuration



This manual applies to the Cortex®-A715 core only. Read this manual together with the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for detailed information about the DSU-110.

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

2.1 Cortex®-A715 core features

You can use the Cortex®-A715 core in a standalone DynamIQ™ configuration, that is, in a homogenous cluster of Cortex®-A715 cores. You can also use the Cortex®-A715 core as the balanced-performance core in a heterogenous cluster.

However, regardless of the cluster configuration, the Cortex®-A715 core has the same features as described in the following lists.

Core features

- Implementation of the Armv9.0-A A64 instruction set
- AArch64 Execution state at all Exception levels, EL0 to EL3
- *Memory Management Unit* (MMU)
- 40-bit *Physical Address* (PA) and 48-bit *Virtual Address* (VA)
- *Generic Interrupt Controller* (GIC) CPU interface to connect to an external interrupt distributor
- Generic Timers interface that supports 64-bit count input from an external system counter
- Implementation of the *Reliability, Availability, and Serviceability* (RAS) Extension
- Implementation of the *Scalable Vector Extension* (SVE) with a 128-bit vector length and *Scalable Vector Extension 2* (SVE2)
- Integrated execution unit with Advanced *Single Instruction Multiple Data* (SIMD) and floating-point support
- Support for the optional Cryptographic Extension

The Cryptographic Extension is licensed separately.

- *Activity Monitoring Unit* (AMU)

Cache features

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

Debug features

- Armv9.0-A debug logic
- *Performance Monitoring Unit* (PMU)

- *Embedded Trace Extension* (ETE)
- *Trace Buffer Extension* (TRBE)
- Optional implementation of the *Statistical Profiling Extension* (SPE)
- Optional *Embedded Logic Analyzer* (ELA), ELA-600


Note

The ELA-600 is licensed separately.

Related information

3. Technical overview on page 31

2.2 Cortex®-A715 core configuration options

You can choose the options that fit your implementation needs at build-time configuration. Configurability is on a per-core basis, that is, in a cluster, various Cortex®-A715 cores can have different configuration options.

You can configure your Cortex®-A715 core implementation using the following options:

Cache protection
: You can configure your implementation with or without cache protection.

SPE
: You can configure your implementation with or without the *Statistical Profiling Extension* (SPE).

PMU counters
: You can configure the number of PMU counters to be 6 or 20.

Cryptographic Extension
: You can configure your implementation with or without the Cryptographic Extension. The Cryptographic Extension is licensed separately.

L1 data cache size
: You can configure the L1 data cache to be 32KB or 64KB.

L1 instruction cache size
: You can configure the L1 instruction cache to be 32KB or 64KB.

L2 cache size
: You can configure the L2 cache to be 128KB, 256KB, or 512KB.

CoreSight™ *Embedded Logic Analyzer* (ELA)
: You can include support for integrating the ELA-600. The ELA-600 is licensed separately.

Timing closure
: You can configure the L2 data cache RAMs timing behavior.

See *RTL configuration process* in the *Arm® Cortex®-A715 Core Configuration and Integration Manual* for detailed configuration options and guidelines.

2.3 DSU-110 dependent features

Support for some *DynamIQ™ Shared Unit-110* (DSU-110) features and behaviors depends on whether your licensed core supports a particular feature.

The following table describes which DSU-110 dependent features are supported in your Cortex®-A715 core.

Table 2-1: Cortex®-A715 core features that have a dependency on the DSU-110

Feature	Supported in the Cortex®-A715 core	Dependency on the DSU-110
Direct connect	No	-
Core included in a complex	No	Affects the DynamIQ™ cluster configuration and external signals.
Cryptographic Extension	Yes, as an option	Affects the external signals of the DSU-110. For more information on MPMM, PDP, and the dispatch block signal, see 5.5 Performance and power management on page 46
Maximum Power Mitigation Mechanism (MPMM)	Yes	
Performance Defined Power (PDP) feature	Yes	
DISPBLKy signal supported	Yes	
Statistical Profiling Extension (SPE) architecture	Yes	
Physical Address (PA) width	40-bit	Affects the CHI and AXI master port bus widths. For more details, see the following parts of the *Arm® DynamIQ™ Shared Unit-110 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®-A715 core implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.5-A. The

Cortex®-A715 core also implements specific Arm®v8-A architecture extensions and supports interconnect, interrupt, timer, debug, and trace architectures.

The Cortex®-A715 core supports AArch64 only at all Exception levels, EL0 to EL3, and supports all mandatory features of each architecture version.

The following tables show, for each Arm®v8-A architecture version, the optional features that the Cortex®-A715 core supports.

Table 2-2: Arm®v8.0-A optional feature support in the Cortex®-A715 core

Feature	Status	Notes
Cryptographic Extension	Supported using a configurable option	See the *Arm® Cortex®-A715 Core Cryptographic Extension Technical Reference Manual* for more technical reference and register information. This extension is licensed separately and access to the documentation is restricted by contract with Arm.
Advanced *Single Instruction Multiple Data* (SIMD) and floating-point support	Supported	See 14. Advanced SIMD and floating-point support on page 86 for more technical reference and register information.
Performance Monitoring Extension (PME)	Supported	See the *Arm® Architecture Reference Manual for A-profile architecture* for information on this feature.

Table 2-3: Arm®v8.1-A optional feature support in the Cortex®-A715 core

Feature	Status	Notes
FEAT_HAFDBS, hardware management of the access flag and dirty state	Supported	See the *Arm® Architecture Reference Manual for A-profile architecture* for information on these features.
FEAT_VMID16, 16-bit VMID	Supported	
FEAT_PAN3	Supported	Enhancement for Privileged Access Never with Execute-only.

Table 2-4: Arm®v8.2-A optional feature support in the Cortex®-A715 core

Feature	Status	Notes
FEAT_HPDS2, Translation Table Page-Based Hardware Attributes	Supported	See the *Arm® Architecture Reference Manual for A-profile architecture* for information on these features.
FEAT_PCSRv8p2, PC Sample-based Profiling	Supported	
FEAT_SHA512 and FEAT_SHA3, SHA2-512 and SHA3 functionality	Supported as part of Armv8-A Cryptographic Extension	
FEAT_SM3 and FEAT_SM4, SM3 and SM4 functionality	Supported as part of Armv8-A Cryptographic Extension	
FEAT_BF16, 16-bit floating-point instructions	Supported	
FEAT_I8MM, Int8 matrix multiply instructions	Supported	
FEAT_SVE, *Scalable Vector Extension* (SVE)	Supported	See 15. Scalable Vector Extensions support on page 87 and the *Arm® Architecture Reference Manual for A-profile architecture* for information on this extension.
FEAT_LPA, Large *Physical Address* (PA) and *Intermediate PA* (IPA) support	Not supported	-

Feature	Status	Notes
FEAT_LVA, Large *Virtual Address* (VA) support	Not supported	-
FEAT_LSMAOC, Load/Store Multiple Atomicity and Ordering Controls	Not supported	-
FEAT_AA32HPD, AArch32 Hierarchical Permission Disables	Not supported	-
FEAT_SPE, *Statistical Profiling Extension* (SPE)	Supported using a configurable option	See 22. Statistical Profiling Extension Support on page 127 and the *Arm® Architecture Reference Manual for A-profile architecture* for information on this optional extension.

Table 2-5: Arm®v8.3-A optional feature support in the Cortex®-A715 core

Feature	Status	Notes
FEAT_NV, Nested Virtualization	Not supported	-
FEAT_CCIDX, Cache extended number of sets	Supported	-
FEAT_PAuth2, pointer authentication enhancements	Supported	-
FEAT_FPAC, *Faulting Pointer Authentication Code* (FPAC)	Supported	-
FEAT_SPEv1p1, Arm®v8.3 Statistical Profiling Extensions	Supported	-

Table 2-6: Arm®v8.4-A optional feature support in the Cortex®-A715 core

Feature	Status	Notes
FEAT_AMUv1, Activity Monitors Extension (AMU)	Supported	See the *Arm® Architecture Reference Manual for A-profile architecture* for information on this feature.
FEAT_MPAM, *Memory System Resource Partitioning and Monitoring* (MPAM) Extension	Supported	See the *Arm® Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A* for information on this extension.
FEAT_NV2, Enhanced support for Nested Virtualization	Not supported	-

Table 2-7: Arm®v8.5-A optional feature support in the Cortex®-A715 core

Feature	Status	Notes
FEAT_MTE, *Memory Tagging Extension* (MTE)	Supported	The Cortex®-A715 core always implements MTE and therefore is compliant with the CHI.E protocol. See *CHI master interface* in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for information on CHI.E commands inferred by MTE.
FEAT_RNG, Random Number Generator Instructions	Not supported	-
FEAT_ExS, Context Synchronization and Exception Handling	Not supported	-
FEAT_MTE2 and FEAT_MTE3, Asymmetric Fault Handling	Supported	MTE enhancement.

Feature	Status	Notes
FEAT_PMUv3p5, PMU Extension version 3.5	Subset of events supported	See 18.1 Performance monitors events on page 98.

The following table shows the Arm®v9.0-A features that the Cortex®-A715 core supports.

Table 2-8: Arm®v9.0-A optional feature support in the Cortex®-A715 core

Feature	Status	Notes
FEAT_SVE2, *Scalable Vector Extension 2* (SVE2)	Supported	See 15. Scalable Vector Extensions support on page 87.
FEAT_ETE, *Embedded Trace Extension* (ETE)	Supported	See 19. Embedded Trace Extension support on page 113.
FEAT_TRBE, *TRace Buffer Extension* (TRBE)	Supported	See 20. Trace Buffer Extension support on page 122
FEAT_SVE_SM4, SVE2 SM4 instructions	Supported	These algorithms are **UNDEFINED** if the Cryptographic Extension is not enabled. See the *Arm® Architecture Reference Manual Supplement, The Scalable Vector Extension* for more information.
FEAT_SVE_SHA3, SVE2 SHA-3 instructions	Supported	
FEAT_SVE_AES, SVE2 AES instructions	Supported	
FEAT_SVE_BitPerm, SVE2 bit permute instructions	Supported	See the *Arm® Architecture Reference Manual Supplement, The Scalable Vector Extension* for more information.
FEAT_TME, *Transactional Memory Extension*	Not supported	-

The following table shows the other standards and specifications that the Cortex®-A715 core supports.

Table 2-9: Other standards and specifications support in the Cortex®-A715 core

Standard or specification	Version	Notes
FEAT_GICv4p1, *Generic Interrupt Controller* (GIC)	GICv4.1	See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information.
Debug	-	Arm®v9.0-A architecture implemented with Arm®v8.4-A Debug architecture support and Arm®v8.3-A Debug over Powerdown support.
CoreSight	v3.0	See the *Arm® CoreSight™ Architecture Specification v3.0* for more information.
FEAT_RASv1p1, *Reliability, Availability, and Serviceability* (RAS)	-	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 77 for more information on the implementation of this extension in the core.
FEAT_ECBHB, *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.

Related information

3.1 Core components on page 31

2.5 Test features

The Cortex®-A715 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®-A715 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.

See *Design for Test integration guidelines* in the *Arm® Cortex®-A715 Core Configuration and Integration Manual* for the list of test signals and information on their usage. See also *Design for Test integration guidelines* in the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for the list of external scan control signals.


Note

The *Arm® Cortex®-A715 Core Configuration and Integration Manual* and *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* are confidential documents that are available with the appropriate product licenses.

2.6 Design tasks

The Cortex®-A715 core is delivered as a synthesizable RTL description in SystemVerilog. Before you can use the Cortex®-A715 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 *RTL configuration process* in the *Arm® Cortex®-A715 Core Configuration and Integration Manual* and in the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for implementation options. See also *Functional integration* in the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for signal descriptions.

2.7 Product revisions

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

Revision	Notes
r0p0	First release
r1p0	SPE functionality supported
r1p1	Added support for FEAT_ECBHB, Exploitative Control using Branch History Buffer information between exception levels
r1p2	Errata fixes

Changes in functionality that have an impact on the documentation also appear in C.1 Revisions on page 671.

3. Technical overview

The components in the Cortex®-A715 core are designed to make it a balanced-performance core.

The main blocks include:

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

The Cortex®-A715 core interfaces with the *DynamIQ™ Shared Unit-110* (DSU-110) through the CPU bridge.

The Cortex®-A715 core implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.5-A.

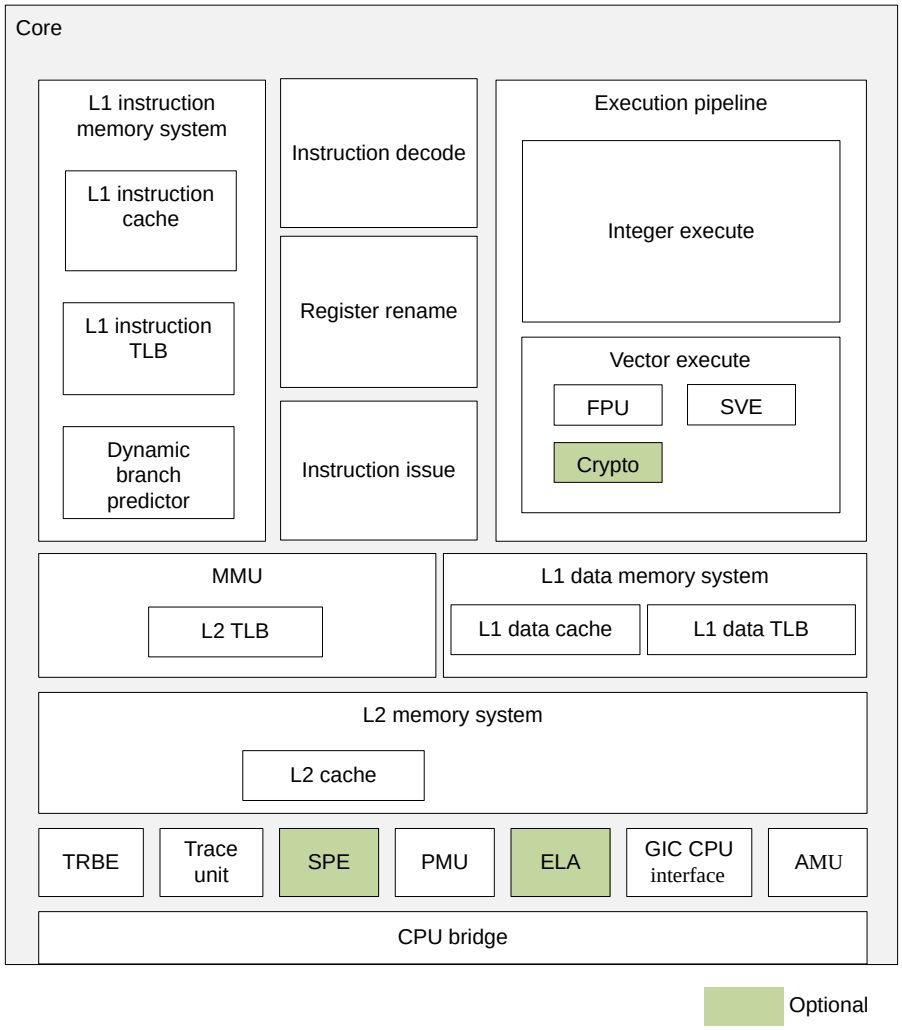
The programmers model and the architecture features implemented, such as the Generic Timer, are compliant with the standards in 2.4 Supported standards and specifications on page 25.

3.1 Core components

The Cortex®-A715 core includes components designed to make it a balanced-performance, low-power, and constrained area product. The Cortex®-A715 core includes a CPU bridge that connects the core to the *DynamIQ™ Shared Unit-110* (DSU-110). The DSU-110 connects the core to an external memory system and the rest of the *System on Chip* (SoC).

The following figure shows the Cortex®-A715 core components.

Figure 3-1: Cortex®-A715 core components



L1 instruction memory system

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

The L1 instruction memory system includes:

- A 32KB or 64KB, 4-way set associative L1 instruction cache with 64-byte cache lines
- A fully associative L1 instruction *Translation Lookaside Buffer* (TLB) with native support for 4KB, 16KB, 64KB, and 2MB page sizes
- A dynamic branch predictor

Instruction decode

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

Register rename

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

Instruction issue

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

Integer execute

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

Vector execute

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

Advanced SIMD and floating-point support

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

Cryptographic Extension

The Cryptographic Extension is optional in the Cortex®-A715 cores. The Cryptographic Extension adds new instructions to the Advanced SIMD and the *Scalable Vector Extension* (SVE) instruction sets that accelerate:

- *Advanced Encryption Standard* (AES) encryption and decryption.
- The *Secure Hash Algorithm* (SHA) functions SHA-1, SHA-3, SHA-224, SHA-256, SHA-384, and SHA-512.
- Armv8.2-SM SM3 hash function and SM4 encryption and decryption instructions.
- Finite field arithmetic that is used in algorithms such as Galois/Counter Mode and Elliptic Curve Cryptography.

The optional Cryptographic Extension is not included in the base product. Arm supplies the Cryptographic Extension only under an additional license to the Cortex®-A715 core license.

Scalable Vector Extension

The *Scalable Vector Extension* (SVE) and *Scalable Vector Extension 2* (SVE2) are extensions to the Armv8-A architecture.
They complement and do not replace AArch64 Advanced SIMD and floating-point functionality.


Note

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

L1 data memory system

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

The L1 data memory system includes:

- A 32KB or 64KB, 4-way set associative cache with 64-byte cache lines.
- A fully associative L1 data TLB with native support for 4KB, 16KB, 64KB, and 2MB page sizes.

Memory Management Unit

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

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

L2 memory system

The L2 memory system includes the L2 cache. The L2 cache is private to the core and is 8-way set associative. You can configure its RAM size to be 128KB, 256KB, or 512KB. The L2 memory system is connected to the DSU-110 through a CPU bridge.

Embedded Trace Extension and Trace Buffer Extension

The Cortex®-A715 core supports a range of debug, test, and trace options including a trace unit and trace buffer.

The Cortex®-A715 core also includes a ROM table that contains a list of components in the system. Debuggers can use the ROM table to determine which CoreSight™ components are implemented.

All the debug and trace components of the Cortex®-A715 core are described in this manual. For more information about the *Embedded Logic Analyzer* (ELA), see the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual*.

Statistical Profiling Extension

The *Statistical Profiling Extension* (SPE) is optional in the Cortex®-A715 cores. The Cortex®-A715 core implements the SPE to the Arm®v8.2-A architecture. The SPE provides a statistical view of the performance characteristics of executed instructions that software writers can use to optimize their code for better performance.

Performance Monitoring Unit

The *Performance Monitoring Unit* (PMU) provides either 6 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®-A715 core implements the Activity Monitors Extension to the Arm®v8.4-A architecture. Activity monitors in the *Activity Monitoring Unit* (AMU) provide useful information for system power management and persistent monitoring.

GIC CPU interface

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

CPU bridge

In a cluster, there is one CPU bridge between each Cortex®-A715 core and the DSU-110.

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

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

Related information

6. Memory management on page 51
7. L1 instruction memory system on page 60
8. L1 data memory system on page 63
9. L2 memory system on page 68
13. GIC CPU interface on page 84
18. Performance Monitors Extension support on page 98
19. Embedded Trace Extension support on page 113
20. Trace Buffer Extension support on page 122
21. Activity Monitors Extension support on page 123
22. Statistical Profiling Extension Support on page 127

3.2 Interfaces

The *DynamIQ™ Shared Unit-110* (DSU-110) manages all Cortex®-A715 core external interfaces to the *System on Chip* (SoC).

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

3.3 Programmers model

The Cortex®-A715 core implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.5-A. The Cortex®-A715 core supports the AArch64 Execution state at all Exception levels, EL0 to EL3.

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

Related information

2.4 Supported standards and specifications on page 25

4. Clocks and resets

To provide dynamic power savings, the Cortex®-A715 core supports hierarchical clock gating. It also supports Warm and Cold resets.

Each Cortex®-A715 core has a single clock domain and receives a single clock input. This clock input is gated by an architectural clock gate in the CPU bridge.

In addition, the Cortex®-A715 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®-A715 core receives the following reset signals from the *DynamIQ™ Shared Unit-110* (DSU-110) side of the CPU bridge:

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

For a complete description of the clock gating and reset scheme of the core, see the following sections in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*:

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

5. Power management

The Cortex®-A715 core provides mechanisms to control both dynamic and static power dissipation.

The dynamic power management includes the following features:

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

The static power management includes the following features:

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

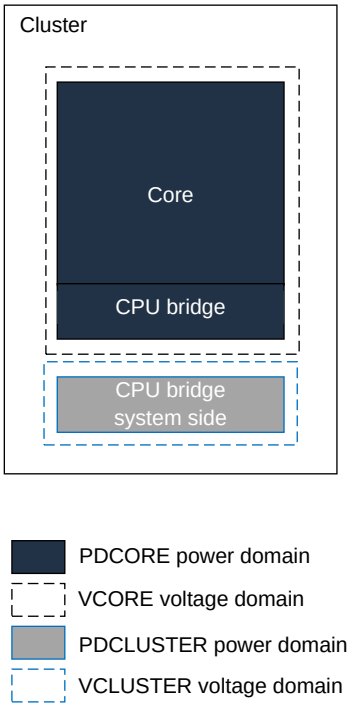
5.1 Voltage and power domains

The *DynamIQ™ Shared Unit-110* (DSU-110) *Power Policy Units* (PPUs) control power management for the Cortex®-A715 core. The core supports one power domain, PDCORE, and one system power domain, PDCLUSTER. Similarly, it supports one core voltage domain, VCORE, and one cluster system voltage domain, VCLUSTER. The power domains and voltage domains have the same boundaries.

The PDCORE power domain contains all Cortex®-A715 core logic and part of the core asynchronous bridge that belongs to the VCORE domain. The PDCLUSTER power domain contains the part of the CPU bridge that belongs to the VCLUSTER domain.

The following figure shows the Cortex®-A715 core power domain and voltage domain. It also shows the cluster power domain and voltage domain that cover the system side of the CPU bridge.

Figure 5-1: Cortex®-A715 core voltage domains and power domains



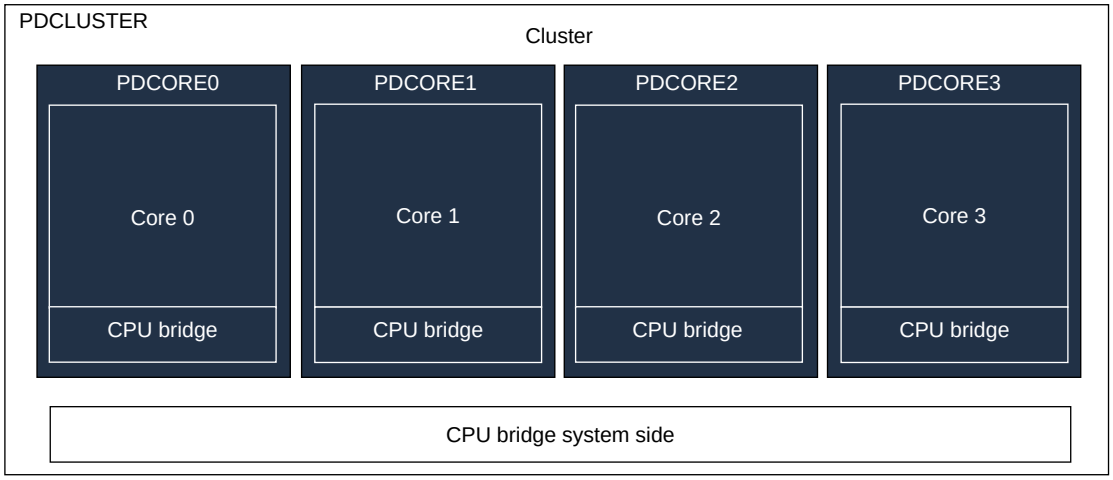
You can tie the VCORE and VCLUSTER voltage domains to the same supply if either:

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

In a cluster with multiple Cortex®-A715 cores, there is one PDCORE<n> power domain per core, where n is the core instance number. If a core is not present, then the corresponding power domain is not present.

This diagram shows the power domains for an example Cortex®-A715 configuration with a four-core cluster:

Figure 5-2: Core power domains in a cluster with four Cortex®-A715 cores



Clamping cells between power domains are inferred through power intent files (UPF) rather than instantiated in the RTL. See *Power management* in the *Arm® Cortex®-A715 Core Configuration and Integration Manual* for more information.

Note

The *Arm® Cortex®-A715 Core Configuration and Integration Manual* is a confidential document that is available with the appropriate product license.

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

5.2 Architectural clock gating modes

The `WFI` and `WFE` instructions put the core into a low-power mode. These instructions 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 the core in a low-power state by disabling most of the core clocks, while keeping the core powered up. When the core is in WFI or WFE state, the input clock is gated externally to the core at the CPU bridge.

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

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


Note

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

The core exits the WFI or WFE state when one of the following 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®-A715 core.

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

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


Note

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

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

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

The PPUs decide and request any change in power mode. The Cortex®-A715 core then performs any actions necessary to reach the requested power mode. For example, the core might gate clocks, clean caches, or disable coherency before it accepts the request.

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

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

5.4 Core power modes

The Cortex®-A715 core power domain has a defined set of power modes and corresponding legal transitions between these modes. The power mode of each core can be independent of other cores in a cluster.

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

The following table shows the supported Cortex®-A715 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 5.6 Cortex-A715 core powerup and powerdown sequence on page 48.

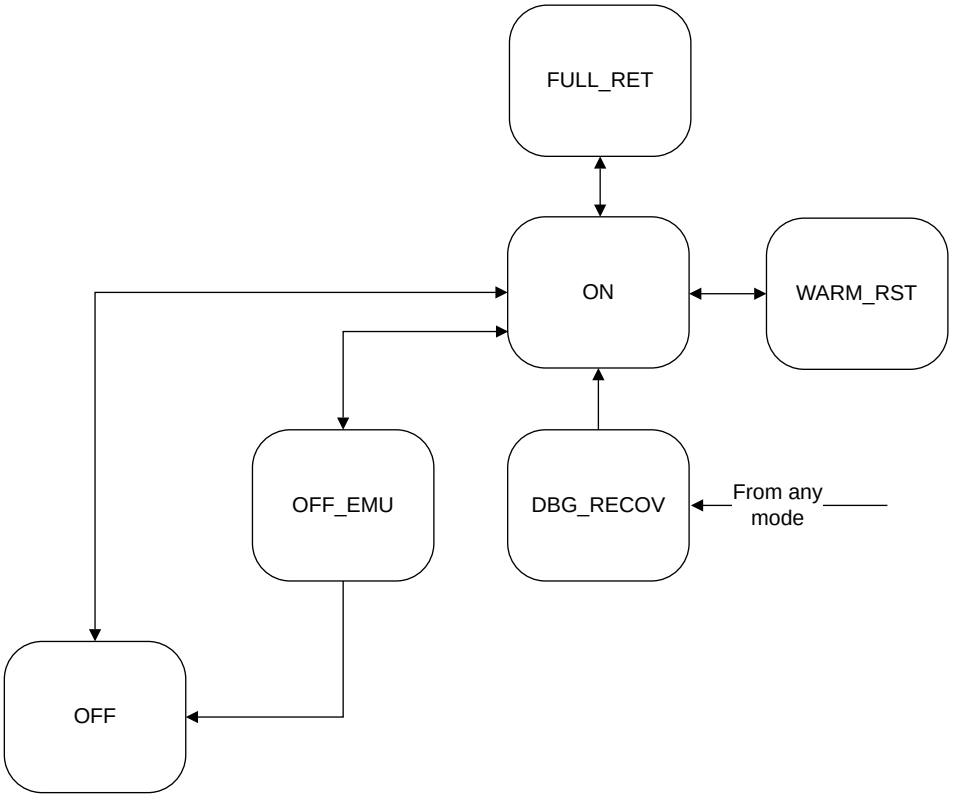
Table 5-1: Cortex®-A715 core power modes

Power mode	Short name	Power state
On	ON	The core is powered up and active.
Full retention	FULL_RET	The core is in retention. 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.

Power mode	Short name	Power state
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 powerdown is normal, 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 cluster, while preserving memory and 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 trace logic and the debug and RAS registers.

The following figure shows the supported modes for the Cortex®-A715 core power domain and the legal transitions between them.

Figure 5-3: Cortex®-A715 core power mode transitions



Related information

5.2 Architectural clock gating modes on page 40
5.4.4 Full retention mode on page 44
5.2.1 Wait for Interrupt and Wait for Event on page 40

5.4.1 On mode

In the On power mode, the Cortex®-A715 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. The L1 and L2 caches are disabled, cleaned, and invalidated, and the core is removed from coherency automatically on transition to Off mode.

A Cold reset can reset the core in this mode.

An attempted external debug access to core debug registers or a utility bus access when the core domain is off returns an error response on the internal debug interface. The error indicates that the core is not available.


Note

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

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 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.1.17 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 170.
- 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 from the DebugBlock of the *DynamIQ™ Shared Unit-110* (DSU)
  - GIC access

5.4.5 Debug recovery mode

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

By default, the core invalidates its caches when it transitions from Off 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. 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 supports preserving the *Reliability, Availability, and Serviceability* (RAS) state. When in Debug recovery mode, a DSU-110 DynamIQ™ cluster-wide Warm reset must be applied externally. The RAS and cache state are preserved when the core is transitioned to the On mode.

Debug recovery 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 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 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

5.4.6 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®-A715 core when the core receives a Warm reset signal from the *DynamIQ™ Shared Unit-110* (DSU-110) side of the CPU bridge.

The Cortex®-A715 core implements the Arm®v8-A Reset Management Register, RMR_EL3. When the core runs in EL3, it requests a Warm reset 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 Performance and power management

The Cortex®-A715 core implements *Performance and Power Management* (PPM) features that can be used to limit high activity events within the core, or trade off efficiency versus peak performance.

The PPM features are:

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

5.5.1 Maximum Power Mitigation Mechanism

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

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

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

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

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

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

Related information

A.2.1 IMP_CPUPPMCR_EL3, Global PPM Configuration Register on page 253
A.4.27 IMP_CPUMPMMCR_EL3, Global MPMM Configuration Register on page 320
B.1.1 CPUPPMCR, Global PPM Configuration Register on page 468
B.1.2 CPUMPMMCR, Global MPMM Configuration Register on page 470

5.5.2 Performance Defined Power

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

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

The PDP has an impact on:

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

- External memory system power reduction. Memory request bandwidth is modulated to reduce power in the memory system.

5.5.3 Dispatch block

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

A pin is provided on the DSU-110 boundary that can directly be used to force the processor to stall for the duration that the pin is asserted. When the processor is stalled, the dispatch of new instructions is stopped. However, instructions that have already been dispatched will continue to execute and complete as normal.

5.6 Cortex®-A715 core powerup and powerdown sequence

There is no specific sequence to power up the Cortex®-A715 core . To power down the core, you must follow a specific sequence. There are no software steps required to bring a core into coherence after reset.

To powerdown the Cortex®-A715 core:

1. If required, save the state of the core to system memory to allow for retrieval of the core state during core powerup.
2. Disable interrupts to the core.
   a. Disable the interrupt enable bits in the ICC_IGRPEN0_EL1 and ICC_IGRPEN1_EL1 registers.
   b. Set the GIC distributor wake-up request for the core using the GICR_WAKER register.
   c. Read the GICR_WAKER register to confirm that the ChildrenAsleep bit indicates that the interface is quiescent.
3. Disable the interrupt outputs from the RAS registers. Alternatively, re-direct the core RAS fault and error interrupt outputs to the system error manager. For more information, see 5.6.1 Managing RAS fault and error interrupts during the core powerdown on page 49.
4. Set the IMP_CPUPWRCTLR_EL1.CORE_PWRDN_EN bit to 1 to indicate to the power controller that a powerdown is requested.
5. Execute an `ISB` instruction.
6. Execute a `WFI` instruction. Once the `WFI` instruction is executed, the powerdown sequence cannot be interrupted.

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

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

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

5.6.1 Managing RAS fault and error interrupts during the core powerdown

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

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

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

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

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

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

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

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

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

If an uncorrectable error occurs during the powerdown, the core remains powered ON and the software remains inactive. The system error manager is then responsible for resetting the entire cluster and the wider system that interacts with the core and cluster. To use this approach, the

system must be designed to allow the core RAS error interrupt to re-route to the system error manager. As the core RAS registers are only accessible to software running on the core, the system error manager is unable to identify where the uncorrectable error occurred within the core.

5.7 Debug over powerdown

The Cortex®-A715 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-110* (DSU-110). The DebugBlock is external to the DSU-110 DynamIQ™ cluster and must remain powered on during the debug over powerdown process.

See *Debug* in the *Arm® DynamIQ™ Shared Unit-110 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®-A715 core internal registers and translation tables. The MMU also controls memory access permissions, memory attributes, 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®-A715 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®-A715 core performs translation table walks for each stage of the translation.

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

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

See the *Arm® Architecture Reference Manual for A-profile architecture* for more information on the *Virtual Memory System Architecture* (VMSA).

6.1 Memory Management Unit components

The Cortex®-A715 core *Memory Management Unit* (MMU) includes several *Translation Lookaside Buffers* (TLBs), an L2 TLB, and a translation table prefetcher.

A TLB is a cache of recently executed page translations within the MMU. The Cortex®-A715 core implements a two-level TLB structure. The L2 TLB stores all page sizes and is responsible for breaking down these pages into smaller pages when required for the L1 data TLB or L1 instruction TLB.

The following table describes the MMU components.

Table 6-1: MMU components

Component	Description
L1 instruction TLB	• Located in the L1 instruction block • Caches entries at the 4KB, 16KB, 64KB, or 2MB granularity of *Virtual Address* (VA) to *Physical Address* (PA) mapping only • Fully associative • 32 entries
L1 data TLB	• Located in the L1 data block • Caches entries at the 4KB, 16KB, 64KB, or 2MB granularity of VA to PA mappings only • Fully associative • 48 entries
L1 *Statistical Profiling Extension* (SPE) TLB	• Located in the SPE block • VA to PA translations of any page and block size • 1 entry
L1 *TRace Buffer Extension* (TRBE) TLB	• Located in the TRBE block • VA to PA translations of any page and block size • 1 entry
L2 TLB	• Located in the MMU block • Includes a walk cache functionality • Made of two translation caches dedicated to specific translation levels: **Small page TLB** ◦ Stores the results of level 3 translations for pages of size 4KB, 16KB, or 64KB ◦ 6-way set associative ◦ 1536 entries **Medium page TLB** ◦ Stores the results of level 2 translations for blocks of size 2MB, 32MB, or 512MB ◦ 4-way set associative ◦ 256 entries
Translation table prefetcher	• Detects access to contiguous translation tables and prefetches the next one • Can be disabled in the ECTLR register

TLB entries contain:

- A global indicator and an *Address Space IDentifier* (ASID) to allow context switches without requiring the TLB to be invalidated
- A *Virtual Machine IDentifier* (VMID) to allow virtual machine switches by the hypervisor without requiring the TLB to be invalidated

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

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

A miss in the L1 data TLB that hits in the L2 TLB has a 5-cycle penalty compared to a hit in the L1 data TLB. This penalty can be increased depending on the arbitration of pending requests.

6.2 TLB entry content

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

Each TLB entry contains:

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

Each TLB entry is associated with either:

- A particular *Address Space IDentifier* (ASID)
- A global indicator

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

Related information

6.4 Translation table walks on page 54

6.3 TLB 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&0
- Non-secure EL2
- Non-secure EL2&0
- Secure EL1&0
- Non-secure EL1&0

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&0 and Non-secure EL1&0 translation regime
- The Secure EL2&0 and Non-secure EL2&0 translation regime

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

- The Secure EL1&0 and Non-secure EL1&0 translation regime, when EL2 is enabled.

6.4 Translation table walks

When the Cortex®-A715 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®-A715 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.

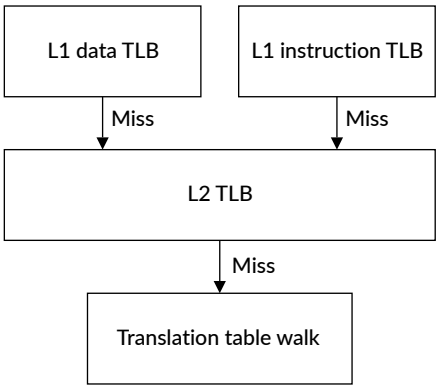
Address translation is performed only when the MMU is enabled. It can also be disabled for a particular translation base register, in which case the MMU returns a Translation Fault.

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 Armv8, for Armv8-A architecture profile* 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.

6.5 Hardware management of the Access flag and dirty state

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

This feature is enabled in TCR_ELx (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®-A715 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®-A715 core returns an abort with the following encoding:

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

6.6 Responses

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

MMU responses

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

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

The responses from the MMU contain the following information:

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

MMU aborts

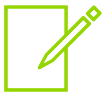
The MMU can detect faults that are related to address translation and can cause exceptions to be taken to the core. Faults can include address size faults, translation faults, access flag faults, and permission faults.

External aborts

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

External aborts are reported synchronously when they occur during:

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


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:

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

Misprogramming contiguous hints

In the case of a misprogramming contiguous hint, 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 stage1 translations
- VTCR_EL2.T0SX for stage2 translations

The Cortex®-A715 core treats such a block as not causing a translation fault and disregards the value of the contiguous bit.

Conflict aborts

The Cortex®-A715 core does not generate conflict abort exceptions.

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

6.7 Memory behavior and supported memory types

The Cortex®-A715 core supports memory types defined in the Armv8-A architecture.

Device memory types have the following attributes:

G – Gathering

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

R – Reordering

The capability to reorder transactions

E – Early Write Acknowledgment

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


Note

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

The following table shows how memory types are supported in the Cortex®-A715 core.

Table 6-2: Supported memory types

Memory attribute type	Shareability	Inner Cacheability	Outer Cacheability	Notes
Device nGnRnE	Outer Shareable	-	-	Treated as Device nGnRnE
Device nGnRE	Outer Shareable[1]	-	-	Treated as Device nGnRE
Device nGRE	Outer Shareable[1]	-	-	Treated as Device nGRE
Device GRE	Outer Shareable[1]	-	-	Treated as Device GRE
Normal	Outer Shareable[1]	Non-cacheable	Any	Treated as Non-cacheable

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

Memory attribute type	Shareability	Inner Cacheability	Outer Cacheability	Notes
Normal	Outer Shareable[1]	Write-Through Cacheable	Any	Treated as Non-cacheable
Normal	Outer Shareable[1]	Write-Back Cacheable	Non-cacheable	Treated as Non-cacheable
Normal	Outer Shareable[1]	Write-Back Cacheable	Write-Through Cacheable	Treated as Non-cacheable
Normal	See Table 6-3: Shareability for Normal memory on page 58.	Write-Back Cacheable (any allocation hint)	Write-Back Cacheable No Allocate	Treated as Write-Back Read and Write Allocate but the outer cacheability propagated to the *DynamIQ™ Shared Unit-110* (DSU-110) is 0 (No Allocate)
Normal	See Table 6-3: Shareability for Normal memory on page 58.	Write-Back Cacheable (any allocation hint)	Write-Back Read or Write Allocate	Treated as Write-Back Read and Write Allocate but the outer cacheability propagated to the DSU-110 is 1, therefore upgraded to Write and Read Allocate

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

Table 6-3: Shareability for Normal memory

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

6.8 Page-based hardware attributes

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

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

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

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

PBHA combination between stage 1 and stage 2 on memory accesses

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

When stage 1 and stage 2 are enabled:

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

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

Enable of PBHA has a granularity of one 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®-A715 core L1 instruction memory system fetches instructions and predicts branches. It includes the L1 instruction cache, the L1 instruction *Translation Lookaside Buffer* (TLB), and the branch prediction unit.

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

The following table shows the L1 instruction memory system features.

Table 7-1: L1 instruction memory system features

Feature	Description
L1 instruction cache	• 32KB or 64KB • 4-way set associative • *Physically Indexed, Physically Tagged* (PIPT) • Optionally protected with parity
Cache line length	64 bytes
Cache policy	*Pseudo-Least Recently Used* (PLRU) cache replacement policy
Interface with L2 memory system	32 bytes per cycle interface


Note

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

7.1 L1 instruction cache behavior

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

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

L1 instruction cache disabled behavior

Disabling the L1 instruction cache has no effect on the operation of the L1 instruction cache. Instructions can be cached into, and fetched from, the L1 instruction cache even when it is disabled. The software must take into account Non-cacheable accesses to ensure correct behavior. See the *Arm® Architecture Reference Manual for A-profile architecture* for more information.

If the L1 instruction cache is disabled, then all memory accesses caused by instruction fetches are performed using the Non-cacheable memory attribute. It means that instruction fetches might not

be coherent with caches in the same core or other cores. The software must take this into account by performing the appropriate cache maintenance operations.

L1 instruction cache maintenance

The cache maintenance operation can happen at any time, regardless of L1I$ status (disabled or enabled).

Related information

5.4.5 Debug recovery mode on page 45

7.2 L1 instruction cache Speculative memory accesses

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

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

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

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

7.3 Program flow prediction

The Cortex®-A715 core contains program flow prediction hardware, also known as branch prediction. Branch prediction increases overall performance and enhances power efficiency.

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

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

Program flow prediction hardware contains the following functionality:

- A *Branch Target Buffer* (BTB) holding the branch target address of previously taken branches

- A *Branch Prediction* (BP) predictor that uses the previous branch history
- The return stack, including nested subroutine return addresses
- A static branch predictor
- An indirect branch predictor

Predicted and non-predicted instructions

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

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

Exception return branch instructions are not predicted.

Return stack

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

Any of the following instructions causes a return stack push:

- `BL`
- `BLR`
- `BLRAA`
- `BLRAAX`
- `BLRAB`
- `BLRABZ`

Any of the following instructions cause a return stack pop:

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

The following instructions are not predicted because they can change the core privilege mode, and the Security state:

- `ERET`
- `ERETAA`
- `ERETAB`

8. L1 data memory system

The Cortex®-A715 core L1 data memory system executes load and store instructions. It services memory coherency requests and specific instructions such as atomics, cache maintenance operations, and memory tagging instructions. The L1 data memory system includes the L1 data cache and the L1 data *Translation Lookaside Buffer* (TLB).

The following table shows the L1 data memory system features.

Table 8-1: L1 data memory system features

Feature	Description
L1 data cache	• 32KB or 64KB • 4-way set associative, 16 banks • *Virtually Indexed, Physically Tagged* (VIPT) behaving as *Physically Indexed, Physically Tagged* (PIPT) • Optionally protected with *Error Correcting Code* (ECC)
Cache line length	64 bytes
Cache policy	Pseudo-*Least Recently Used* (LRU) cache replacement policy
Interface with integer execute pipeline and vector execute	• 3×64-bit read paths and 4×64-bit write paths for the integer execute pipeline • 2×128-bit write paths and 3×128-bit read paths for the vector execute


Note

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

8.1 L1 data cache behavior

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

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

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

L1 data cache disabled behavior

If the L1 data cache is disabled, then:

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

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

To maintain data coherency between multiple cores, the Cortex®-A715 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. This operation is compliant with the Armv8-A architecture.

Related information

5.4.5 Debug recovery mode on page 45

8.2 Write streaming mode

The Cortex®-A715 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 *Merge Write Buffer* (MWB) detects when the core has written a full cache line before the linefill completes. If this situation is detected on a configurable number of consecutive linefills, then the MWB switches into write streaming mode.

When in write streaming mode, load operations behave as normal, and can still cause linefills. When in write streaming mode, write operations 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 master interface before the MWB 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 subsequent load operation that targets the same line as an outstanding write stream.

When a Cortex®-A715 core has switched to write streaming mode, the MWB 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) by writing the register A.1.15 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 159.

8.3 Instruction implementation in the L1 data memory system

The Cortex®-A715 core supports the atomic instructions added in the Arm®v8.1-A architecture.

Atomic instructions to Cacheable memory can be performed as either near atomics or far atomics, the Cortex®-A715 core performs these instructions as near atomics by default.

Alternatively, CPUECTLR can be programmed so that depending on the system behavior, some atomic instructions attempt to execute as far atomics.

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

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

The Cortex®-A715 core supports atomics to Device or Non-cacheable memory, however this relies on the interconnect also supporting atomics. If such an atomic instruction is executed when the interconnect does not support them, then it results in an abort.

8.4 Internal exclusive monitor

The Cortex®-A715 core includes an internal exclusive monitor with a 2-state, open and exclusive state machine that manages Load-Exclusive and Store-Exclusive accesses and Clear-Exclusive (CLREX) instructions.

You can use these instructions to construct semaphores, ensuring synchronization between different processes running on the core, and also between different cores that are using the same coherent memory locations for the semaphore. A Load-Exclusive instruction tags a small block of memory for exclusive access. CTR_EL0 defines the size of the tagged blocks as 16 words, one cache line.


Note

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

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

See A.4.23 CTR_EL0, Cache Type Register on page 312 for more technical reference and register information.

8.5 Data prefetching

Data prefetching fetches data before it is needed to boost execution performance.

For cases that cannot be handled efficiently by data prefetchers, the Cortex®-A715 core supports the AArch64 prefetch memory instructions, PRFM.

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

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

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

Hardware data prefetcher

The load/store unit includes hardware prefetcher engines that are responsible for generating prefetches targeting the L1, L2, and L3 caches. Specifically, the prefetch engine in the L1 memory subsystem targets the L1 and L2 cache. The prefetch engine in the L2 memory subsystem targets

the L2 and L3 cache. The load side prefetcher uses the *Virtual Address* (VA) and the *Program Counter* (PC). The store side prefetcher uses the *Virtual Address* (VA) only.

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

- A.1.15 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 159
- A.1.16 IMP_CPUECTLR2_EL1, CPU Extended Control Register on page 167

Data cache zero

In the Cortex®-A715 core, the *Data Cache Zero by Virtual Address* (`DC ZVA`) instruction enables a block of 64 bytes in memory, aligned to 64 bytes in size, to be set to zero.

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

9. L2 memory system

The Cortex®-A715 core L2 memory system connects the core with the *DynamIQ™ Shared Unit-110* (DSU-110) through the CPU bridge. It includes the private L2 cache.

The L2 cache is unified and private to each Cortex®-A715 core in a cluster.

The L2 memory system includes data prefetcher engines that use the *Virtual Address* (VA) and the *Program Counter* (PC). The different engines are able to prefetch data in the L2 cache and the L3 cache.

The following table shows the L2 memory system features.

Table 9-1: L2 memory system features

Feature	Description
L2 cache	• 128KB, 256KB, or 512KB • 8-way set associative, 2 banks • *Physically Indexed, Physically Tagged* (PIPT) • Optionally protected with with *Error Correcting Code* (ECC)
Cache line length	64 bytes
Cache policy	Dynamic biased cache replacement policy
Interface with the *DynamIQ™ Shared Unit-110* (DSU-110)	One CHI Issue E compliant interfaces with 256-bit read and write channel widths

9.1 L2 cache

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

The L1 instruction cache and L2 cache are weakly inclusive. Instruction fetches that miss in the L1 instruction cache and L2 cache allocate both caches, but the invalidation of the L2 cache does not cause back-invalidates of the L1 instruction cache.

The L1 data cache and L2 cache are strictly exclusive. Any data contained in the L1 data cache is never present in the L2 cache.

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


Note

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

over the number of sets and ways described in CCSIDR_EL1 for that cache. This operation is compliant with the Armv8-A architecture.

Related information

5.4.5 Debug recovery mode on page 45

9.2 Support for memory types

The Cortex®-A715 core simplifies coherency logic by downgrading some memory types.

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

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

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

The additional attribute hints are used as follows:

Allocation hint

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

Transient hint

All cacheable reads and writes that have the transient bit set allocate in the L2 cache.

An allocating read to the L1 data cache that has the transient bit set is allocated in the L1 cache. Transient lines evicted from the L1 cache do not allocate downstream caches.

9.3 Transaction capabilities

The CHI Issue E interface between the Cortex®-A715 core L2 memory system and the *DynamIQ™ Shared Unit-110* (DSU-110) provides transaction capabilities for the core.

The following table shows the maximum possible values for read, write, *Distributed Virtual Memory* (DVM) issuing, and snoop capabilities of the Cortex®-A715 core L2 cache.

Table 9-2: Cortex®-A715 core transaction capabilities

Attribute	Maximum value	Description
Write issuing capability	120	This is the maximum number of outstanding write transactions for memory that is cacheable, Non-cacheable, and Device GRE/nGRE.
	30	This is the maximum number of outstanding write transactions for memory that is Device nGnRE and nGnRnE.

Attribute	Maximum value	Description
Read issuing capability	60	This is the maximum number of outstanding read transactions for memory that is cacheable, Non-cacheable, and Device GRE/nGRE.
	30	This is the maximum number of outstanding read transactions for memory that is Device nGnRE and nGnRnE.
Snoop acceptance capability	64	This is the maximum number of outstanding snoops accepted.
DVM issuing capability	30	This is the 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®-A715 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.

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

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

You can access the contents of the internal memory using the twelve read-only (RO) System registers in Table 10-1: System registers used to access internal memory on page 71. The internal memory is selected by programing the **IMPLEMENTATION DEFINED** RAMINDEX register using the following `SYS` instruction:

```
SYS #6, C15, C0, #0, <Xt>
```

For more information on the RAMINDEX register, see A.3.1 RAMINDEX, RAMINDEX system instruction on page 255. The data is read from the read-only System registers as shown in the following table.

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

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

Register name	Function	Operation
IMP_ISIDE_DATA0_EL3	Instruction Data register 0	`MRS <Xt>, S3_6_c15_c0_0`
IMP_ISIDE_DATA1_EL3	Instruction Data register 1	`MRS <Xt>, S3_6_c15_c0_1`
IMP_ISIDE_DATA2_EL3	Instruction Data register 2	`MRS <Xt>, S3_6_c15_c0_2`
IMP_DSIDE_DATA0_EL3	L1D Data register 0	`MRS <Xt>, S3_6_c15_c1_0`
IMP_DSIDE_DATA1_EL3	L1D Data register 1	`MRS <Xt>, S3_6_c15_c1_1`
IMP_DSIDE_DATA2_EL3	L1D Data register 2	`MRS <Xt>, S3_6_c15_c1_2`
IMP_L2_DATA0_EL3	L2 Data register 0	`MRS <Xt>, S3_6_C15_C1_3`
IMP_L2_DATA1_EL3	L2 Data register 1	`MRS <Xt>, S3_6_C15_C1_5`

Register name	Function	Operation
IMP_L2_DATA2_EL3	L2 Data register 2	`MRS <Xt>, S3_6_C15_C1_4`
IMP_MMU_DATA0_EL3	TLB Data register 0	`MRS <Xt>, S3_6_C15_C0_3`
IMP_MMU_DATA1_EL3	TLB Data register 1	`MRS <Xt>, S3_6_C15_C0_4`
IMP_MMU_DATA2_EL3	TLB Data register 2	`MRS <Xt>, S3_6_C15_C0_5`

10.1 L1 cache encodings

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

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

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

Table 10-2: Cortex®-A715 L1 instruction cache tag location encoding for 64KB

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

Table 10-3: Cortex®-A715 L1 instruction cache tag location encoding for 32KB

Bit field of Xn	Description
[31:24]	RAMID = `0x00`
[23:20]	Reserved
[19:18]	Way
[17:13]	Reserved
[12:6]	Virtual Address bits[12:6]
[5:0]	Reserved

Table 10-4: Cortex®-A715 L1 instruction cache data location encoding for 64KB

Bit field of Xn	Description
[31:24]	RAMID = `0x01`
[23:20]	Reserved
[19:18]	Way
17	Reserved
[16:14]	Virtual Address bits[5:3]
[13:6]	Virtual Address bits[13:6]

Bit field of Xn	Description
[5:0]	Reserved

Table 10-5: Cortex®-A715 L1 instruction cache data location encoding for 32KB

Bit field of Xn	Description
[31:24]	RAMID = `0x01`
[23:20]	Reserved
[19:18]	Way
17	Reserved
[16:14]	Virtual Address bits[5:3]
13	Reserved
[12:6]	Virtual Address bits[12:6]
[5:0]	Reserved

Table 10-6: Cortex®-A715 L1 data cache tag location encoding for 64KB (<n> = 13), and 32KB (<n> = 12)

Bit field of Xn	Description
[31:24]	RAMID = `0x08`
[23:20]	Reserved
[19:18]	Way
[17:16]	Bank selection **`0b00`** Tag RAM 0 **`0b01`** Tag RAM 1 **`0b10`** Tag RAM 2
[15:<n+1>]	Unused
[<n>:6]	Virtual Address bits[<n>:6]
[5:0]	Reserved

Table 10-7: Cortex®-A715 L1 data cache data location encoding for 64KB (<n> = 13), and 32KB (<n> = 12)

Bit field of Xn	Description
[31:24]	RAMID = `0x09`
[23:20]	Reserved
[19:18]	Way
[17:16]	Virtual Address bits[5:4]
[15:<n+1>]	Unused
[<n>:6]	Virtual Address bits[<n>:6]
[5:0]	Reserved

10.1.1 L1 RAM returned data

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

Click the register name in the following table for details on the returned data format.

Table 10-8: Generic system control register summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_ISIDE_DATA0_EL3	3	6	C15	C0	0	–	64-bit	RAMINDEX Instruction Data register 0
IMP_ISIDE_DATA1_EL3	3	6	C15	C0	1	–	64-bit	RAMINDEX Instruction Data register 1
IMP_ISIDE_DATA2_EL3	3	6	C15	C0	2	–	64-bit	RAMINDEX Instruction Data register 2
IMP_DSIDE_DATA0_EL3	3	6	C15	C1	0	–	64-bit	RAMINDEX L1D Data register 0
IMP_DSIDE_DATA1_EL3	3	6	C15	C1	1	–	64-bit	RAMINDEX L1D Data register 1
IMP_DSIDE_DATA2_EL3	3	6	C15	C1	2	–	64-bit	RAMINDEX L1D Data register 2

10.2 L2 cache encodings

The L2 cache is 8-way set associative.

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

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

Table 10-9: Cortex®-A715 L2 cache tag location encoding for 512KB (<n> = 15), 256KB (<n> = 14), and 128KB (<n> = 13)

Bit field of Xn	Description
[31:24]	RAMID = `0x10`
[23:21]	Reserved
[20:18]	Way
[17:<n>+1]	Reserved
[<n>:7]	Physical Address bits[<n>:7]
[6]	Superbank - Physical Address bit[6]
[5:0]	Reserved

Table 10-10: Cortex®-A715 L2 cache data location encoding for 512KB (<n> = 15), 256KB (<n> = 14), and 128KB (<n> = 13)

Bit field of Xn	Description
[31:24]	RAMID = `0x11`
[23:21]	Reserved
[20:18]	Way
[17:<n>+1]	Reserved

Bit field of Xn	Description
[<n>:7]	Physical Address bits[<n>:7]
[6]	Superbank - Physical Address bit[6]
[5:4]	16B granule inside the line
[3:0]	Reserved

10.2.1 L2 RAM returned data

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

Click the register name in the following table for details on the returned data format.

Table 10-11: Generic system control register summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_L2_DATA0_EL3	3	6	C15	C1	3	—	64-bit	RAMINDEX L2 Data register 0
IMP_L2_DATA2_EL3	3	6	C15	C1	4	—	64-bit	RAMINDEX L2 Data register 2
IMP_L2_DATA1_EL3	3	6	C15	C1	5	—	64-bit	RAMINDEX L2 Data register 1

10.3 L2 TLB encodings

The L2 TLB RAM for small pages (TCSP) is 6-way set associative, and the L2 TLB RAM for medium pages (TCMP) is 4-way set associative.

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

Table 10-12: Cortex®-A715 L2 TLB location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x18`
[23:20]	Reserved
[19:16]	• TCSP: Way (0-5) • TCMP: Way (0-3)
[15:13]	Reserved
12	Array **0b0** TCSP **0b1** TCMP
[11:8]	Reserved
[7:0]	• TCSP: Index (0-255) • TCMP: Index (0-63)

10.3.1 L2 TLB RAM returned data

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

Click the register name in the following table for details on the returned data format.

Table 10-13: Generic system control register summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_MMU_DATA0_EL3	3	6	C15	C0	3	–	64-bit	RAMINDEX TLB Data register 0
IMP_MMU_DATA1_EL3	3	6	C15	C0	4	–	64-bit	RAMINDEX TLB Data register 1
IMP_MMU_DATA2_EL3	3	6	C15	C0	5	–	64-bit	RAMINDEX TLB Data register 2

11. RAS Extension support

The Cortex®-A715 core supports the *Reliability, Availability, and Serviceability* (RAS) v1.1 Extension, however it does not support the optional RAS Timestamp Extension.

In particular, the Cortex®-A715 core supports these RAS Extension features:

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

The Cortex®-A715 core features the following nodes:

- Node 0 that includes the shared L3 memory system in the *DynamIQ™ Shared Unit-110* (DSU-110)
- Node 1 that includes the private L1 and L2 memory systems in the core

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

For information on the node that includes the shared L3 memory system, see *RAS Extension support* in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*.

11.1 Cache protection behavior

The configuration of the *Reliability, Availability, and Serviceability* (RAS) Extension that is implemented in the Cortex®-A715 core includes cache protection. In this case, the Cortex®-A715 core protects against errors that result in a RAM bitcell holding the incorrect value.

The RAMs in the Cortex®-A715 core have the following capabilities:

SED parity

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

The following table indicates which protection type is applied to each RAM. 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	Parity or ECC support
L1 instruction cache tag	SED parity
L1 instruction cache data	
Translation Lookaside Buffer (TLB)	
L1 data cache tag	SECDED ECC
L1 data cache auxiliary tag	
L1 data cache data	
L2 cache tag	
L2 cache data	
L2 *Transaction Queue* (TQ)	

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. This might cause data corruption.

If there are three or more bit errors within the same protection granule, the core might or might not detect the errors. Whether it 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®-A715 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®-A715 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. Core node errors are reported using nCOREFAULTIRQ.

Uncorrected faults

When ERR1CTLR.FI is set, all detected Deferred errors and Uncorrected errors that core n detects generate an FHI through the nFAULTIRQ[n+1] signal.

When ERR1CTLR.CFI or any other CE-counter overflow bits are set, then all detected Corrected errors also cause an FHI to be generated.

Uncorrected interrupts

When ERR1CTLR.UI is set, all detected Uncorrected errors that are not deferred that core n detects generate an error recovery interrupt on the nCOREERRIQ signal.

Clearing reported faults

nFAULTIRQ[n+1] and nERRIRQ[n+1] signals must remain asserted until software clears them by writing the ERR1STATUS register.

11.4 Error detection and reporting

When the Cortex®-A715 core consumes an error, it raises different exceptions depending on the error type.

The Cortex®-A715 core might raise:

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

Error detection and reporting registers

The following registers are provided:

- Error Record Feature Registers, ERR1FR. These read-only registers specify various error record settings.
- Error Record Control Registers, ERR1CTLR. These registers enable error reporting and also enable various interrupts that are related to errors and faults.
- Error Record Miscellaneous Registers, ERR1MISC0-3. These registers record details of the error location and counts.
- Pseudo-fault Generation Feature register, ERR1PFGF. This read-only register specifies various error settings.

11.4.1 Error reporting and performance monitoring

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

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

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

Related information

18.1 Performance monitors events on page 98

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®-A715 core can inject the following error types:

Corrected errors

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

Deferred errors

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

Uncontainable errors

An *Uncontainable Error* (UC) is generated for a double ECC error on the L1 tag RAM, or L2 tag RAM following an eviction.

An error can be injected immediately or when a 32-bit counter reaches zero. You can control the value of the counter through the Error Pseudo-fault Generation Countdown Register, ERR1PFGCDN. The value of the counter decrements on a per clock cycle basis. See the *Arm®*

Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile for more information about ERR1PFGCDN.


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 **IMPLEMENTATION DEFINED** RAS registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

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

12. Utility bus

The utility bus provides access to control registers for various system components in the *DynamIQ™ Shared Unit-110* (DSU-110) and the cores within the DSU-110 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®-A715 core:

- *Activity Monitor Unit* (AMU) registers in the cores
- *Max Power Mitigation Mechanism* (MPMM) registers in the cores

Information about the PPU registers for the cores in the cluster is provided in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*. For all other registers accessed by the utility bus see *Utility bus* in the *Arm® DynamIQ™ Shared Unit-110 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.

- The base address for each set of registers for the core AMU, and MPMM registers depend on the core instance number <n>, from 0 to the total number of cores minus one.
- In the following table, any address space that is not documented is treated as **RAZ/WI**.
- The base addresses in the following table are the addresses accessed on the utility bus interface. The system interconnect typically maps these addresses into a particular address range based on the system address map. Therefore, software has to add the base address listed here onto the system address range base to get the absolute physical address of a register.

Table 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.4 External AMU registers summary on page 572
0x<n>B_0000	Core <n> MPMM	Secure	B.1 External MPMM registers summary on page 468
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-110 Technical Reference Manual*.

13. GIC CPU interface

The *Generic Interrupt Controller* (GIC) supports and controls interrupts. The GIC Distributor connects to the Cortex®-A715 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-110 DynamIQ™ cluster has a GIC CPU interface, which connects to a common external distributor component.

The GICv4.1 architecture implemented in the Cortex®-A715 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®-A715 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®-A715 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-110 Configuration and Integration Manual* for more information on these signals.

13.2 AArch64 GIC system registers

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table 13-1: GIC system registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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_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
ICH_VTR_EL2	3	4	C12	C11	1	–	64-bit	Interrupt Controller VGIC Type Register
ICC_CTLR_EL3	3	6	C12	C12	4	–	64-bit	Interrupt Controller Control Register (EL3)

14. Advanced SIMD and floating-point support

The Cortex®-A715 core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set without floating-point exception trapping.

The Cortex®-A715 core floating-point implementation includes features up to Arm®v8.5-A. BFloat16 floating-point and Int8 matrix multiplication are part of these supported features.

The Cortex®-A715 core implements all operations in hardware with support for all combinations of:

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

15. Scalable Vector Extensions support

The Cortex®-A715 core supports the *Scalable Vector Extension* (SVE) and the *Scalable Vector Extension 2* (SVE2). SVE and SVE2 complement and do not replace AArch64 Advanced SIMD and floating-point functionality.

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

The Cortex®-A715 core implements a scalable vector length of 128 bits.

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

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.

17. Debug

The DSU-110 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 DynamIQ™ cluster, and others in the separate DebugBlock.

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

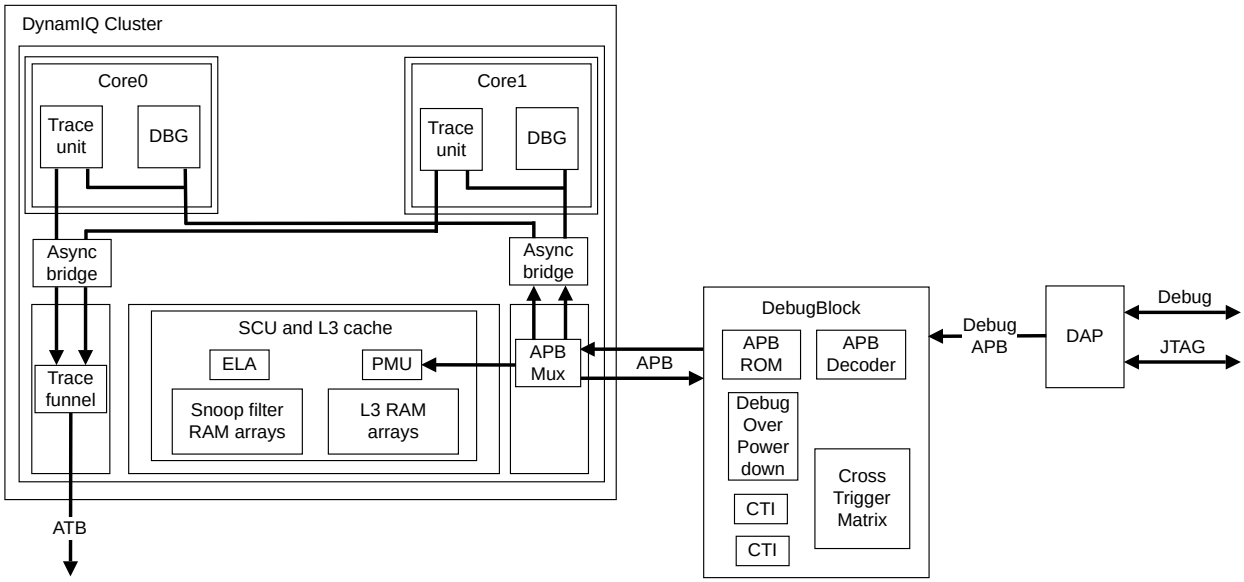
The connection between the cluster and the DebugBlock consists of a pair of *Advanced Peripheral Bus* APB interfaces, one in each direction. All debug traffic, except the authentication interface, takes place over this interface as read or write APB transactions. This debug traffic includes register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

Figure 17-1: 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 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 DynamIQ™ cluster down to a single AMBA® 4 ATBv1.1 interface.

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

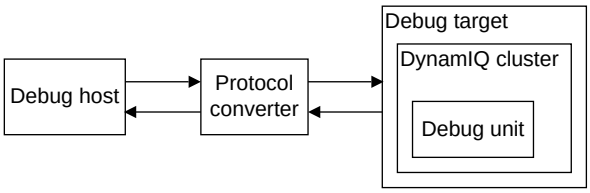
The Cortex®-A715 core also supports direct access to internal memory, that is, cache debug. Direct access to internal memory allows software to read the internal memory that the L1 and L2 cache and *Translation Lookaside Buffer* (TLB) structures use. See 10. Direct access to internal memory on page 71 for more information.

17.1 Supported debug methods

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

The following figure shows a typical external debug system.

Figure 17-2: External debug system



Debug host

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

Protocol converter

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

Debug target

The lowest level of the system implements system support for the protocol converter to access the debug unit. For DSU-110 based devices, the mechanism used to access the debug unit is based on the CoreSight architecture. The DSU-110 DebugBlock is accessed using an APB interface and the debug accesses are then directed to the selected A715 core inside the DynamIQ™ cluster. An example of a debug target is a development system with a test chip or a silicon part with a A715 core.

Debug unit

Helps debugging software that is running on the core:

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

With the debug unit, you can:

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

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

17.2 Debug register interfaces

The Cortex®-A715 core implements the Arm®v9.0-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-110 Technical Reference Manual* for more information.

Related information

5.7 Debug over powerdown on page 50

17.2.1 Core interfaces

System register access allows the Cortex®-A715 core to access certain debug registers directly. The debug register interfaces provide access to these registers either from software running on the core or from an external debugger.

Access to the debug registers is partitioned as follows:

Debug

This function is both system register based and memory-mapped. You can access the debug register map using the APB slave port that connects into the DebugBlock of the *DynamIQ™ Shared Unit-110* (DSU-110).

Performance monitoring

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

Trace

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

Statistical profiling

This function is system register based.

ELA registers

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

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

17.2.2 Effects of resets on debug registers

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

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

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

17.2.3 External access permissions to Debug registers

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

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

Table 17-1: External access conditions to registers

Name	Condition	Description
Off	EDPRSR.PU = 1	Because Armv8.3-DoPD, Debug over PowerDown, is implemented, access to this field is *Read-As-One* (RAO). When the core power domain is in a powerup state, the Debug registers in the core power domain can be accessed. When the core power domain is OFF, accesses to the Debug registers in the core power domain, including EDPRSR, return an error.
OSLK	OSLSR_EL1.OSLK = 1	OS Lock is locked.

Name	Condition	Description
EDAD	`AllowExternalDebugAccess()==FALSE`	External debug access is disabled. If an error is returned because of an EDAD condition code, and this is the highest priority error condition, then EDPRSR.SDAD is set to 1. Otherwise, SDAD is unchanged.
Default	-	This is normal access, none of the conditions apply.

17.2.4 Breakpoints and watchpoints

The Cortex®-A715 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®-A715 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®-A715 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-110 DynamIQ™ cluster level.

See *Debug* and *ROM tables* in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*.

17.5 ROM table

The Cortex®-A715 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®-A715 core. There is one ROM table for each core and ROM tables comply with the *Arm® CoreSight™ Architecture Specification v3.0*.

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

The Cortex®-A715 core ROM table includes the following entries:

Table 17-2: Core ROM table

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

Related information

17.9 External ROM table registers on page 96

17.6 CoreSight™ component identification

Each component associated with the Cortex®-A715 core has a unique set of CoreSight™ ID values. The following table shows these values.

Table 17-3: Cortex®-A715 core CoreSight™ component identification

Component	Peripheral ID	Component ID	DevType	DevArch	Core revision
DBG	0x04201BBD4D	0xB105900D	0x15	0x47709A15	r1p2
PMU			0x16	0x47702A16	
Trace unit			0x13	0x47705A13	
ROM table			0x00	0x47700AF7	

17.7 CTI register identification values

The Cortex®-A715 core CTI registers are located in the DebugBlock of the DSU-110.

For the cluster and core CTI register names and descriptions, see *External CTI register summary* in the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual*. Only the core CTI register peripheral ID values will differ from the cluster CTI register peripheral ID values.

The following table shows the core CTI register peripheral ID values.

Table 17-4: Core CTI register peripheral ID values

Register	Bitfield position	Bitfield name	Value
CTIPIDR4	[7:4]	SIZE	See register B.2.37 PMPIDR4, Performance Monitors Peripheral Identification Register 4 on page 528
	[3:0]	DES_2	
CTIPIDR3	[7:4]	REVAND	See register B.2.41 PMPIDR3, Performance Monitors Peripheral Identification Register 3 on page 534
	[3:0]	CMOD	
CTIPIDR2	[7:4]	REVISION	See register B.2.40 PMPIDR2, Performance Monitors Peripheral Identification Register 2 on page 532
	[3]	JEDEC	
	[2:0]	DES_1	
CTIPIDR1	[7:4]	DES_0	See register B.2.39 PMPIDR1, Performance Monitors Peripheral Identification Register 1 on page 531
	[3:0]	PART_1	
CTIPIDR0	[7:0]	PART_0	See register B.2.38 PMPIDR0, Performance Monitors Peripheral Identification Register 0 on page 530

17.8 External Debug registers

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table 17-5: Debug registers summary

Offset	Name	Reset	Width	Description
0x090	EDRCR	–	32-bit	External Debug Reserve Control Register
0x094	EDACR	–	32-bit	External Debug Auxiliary Control Register
0x310	EDPRCR	–	32-bit	External Debug Power/Reset Control Register
0xD00	MIDR_EL1	–	32-bit	Main ID Register

Offset	Name	Reset	Width	Description
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
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

17.9 External ROM table registers

The summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped ROM table registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table 17-6: 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
0xFB8	AUTHSTATUS	—	32-bit	Authentication Status Register
0xFBC	DEVARCH	—	32-bit	Device Architecture Register
0xFD0	PIDR4	—	32-bit	Peripheral Identification Register 4
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

Offset	Name	Reset	Width	Description
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®-A715 core implements the Performance Monitors Extension, including Arm®v8.4-A and Arm®v8.5-A performance monitoring features.

The Cortex®-A715 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®-A715 core *Performance Monitoring Unit* (PMU) collects events from other units in the design and uses numbers to reference these events.

CommonEvent PMU events

The following table shows the Cortex®-A715 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.


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: CommonEvent PMU events

Event number	Mnemonic	Description
0x0000	SW_INCR	Instruction architecturally executed, Condition code check pass, software increment This event counts any instruction architecturally executed (condition code check pass).

Event number	Mnemonic	Description
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 L1 instruction TLB from the MMU Translation Cache (MMUTC). This includes refills that result in a translation fault. The following instructions are not counted: • TLB maintenance instructions This event counts regardless of whether the 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 L1. 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. • 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.
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.

Event number	Mnemonic	Description
0x0008	INST_RETIRED	Instruction architecturally executed This event counts all retired instructions, including ones 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
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.
0x000D	BR_IMMED_RETIRED	Instruction architecturally executed, immediate branch 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	Instruction architecturally executed, Condition code check pass, procedure return
0x0010	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed This event counts any predictable branch instruction which is mispredicted either due to dynamic misprediction or because the MMU is off and the branches are statically predicted not taken.
0x0011	CPU_CYCLES	Cycle
0x0012	BR_PRED	Predictable branch 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. The following instructions are not counted: • 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.

Event number	Mnemonic	Description
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 or L3. 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. • Full-line writes which write to L2 without writing L1, such as write-streaming mode.
0x0016	L2D_CACHE	Level 2 data cache access • If the core is configured with a per-core L2 cache, this event counts any transaction from L1 which looks up in the L2 cache, and any writeback from the L1 to the L2. Snoops from outside the core and cache maintenance operations are not counted. • If the core is not configured with a per-core L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE. • If neither a per-core cache nor a cluster cache are configured, then this event is not implemented.
0x0017	L2D_CACHE_REFILL	Level 2 data cache refill • If the core is configured with a per-core L2 cache, this event counts any Cacheable transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 are not meant to be counted. • If the core is not configured with a per-core L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE_REFILL. • If neither a per-core cache nor a cluster cache are configured, then this event is not implemented.
0x0018	L2D_CACHE_WB	Level 2 data cache write-back If the core is configured with a per-core L2 cache, this event counts any write-back of data from the L2 cache to a location outside the core. The event includes snoops to the L2 which return data, regardless of whether they cause an invalidation. Invalidations from the L2 which do not write data outside of the core and snoops which return data from the L1 are not counted. • If the core is not configured with a per-core L2 cache, this event is not implemented.
0x0019	BUS_ACCESS	This event counts for every beat of data that is transferred over the data channels between the core and the SCU. If both read and write data beats are transferred on a given cycle, this event is counted twice on that cycle. This event counts the sum of BUS_ACCESS_RD and BUS_ACCESS_WR.
0x001B	INST_SPEC	Operation speculatively executed This event duplicates INST_RETIRED.
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.

Event number	Mnemonic	Description
0x001D	BUS_CYCLES	Bus cycle This event duplicates CPU_CYCLES.
0x001E	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.
0x0020	L2D_CACHE_ALLOCATE	Level 2 data cache allocation without refill This event counts any full cache line write into the L2 cache which does not cause a linefill, including write-backs from L1 to L2 and full-line writes which do not allocate into L1.
0x0021	BR_RETIRED	Instruction architecturally executed, branch This event counts all branches, taken or not, popped from the branch monitor. This excludes exception entries, debug entries, and CCFAIL branches. In the ** core, an ISB is a branch, and even micro architectural ISBs are counted.
0x0022	BR_MIS_PRED_RETIRED	Instruction architecturally executed, mispredicted branch This event counts any branch that is counted by BR_RETIRED which is not correctly predicted and causes a pipeline clean.
0x0023	STALL_FRONTEND	No operation has been issued, because of 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 has been issued, because of 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 data L1 TLB. If both a load and a store are executed on a cycle, this event counts twice. This event counts regardless of whether the MMU is enabled.
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.
0x0029	L3D_CACHE_ALLOCATE	Attributable level 3 data or unified cache allocation without refill This event counts any full cache line write into the L3 cache which does not cause a linefill, including write-backs from L2 to L3 and full-line writes which do not allocate into L2. This event also counts WriteUnique and datafull WriteEvictOrEvict.
0x002A	L3D_CACHE_REFILL	Attributable level 3 data or unified cache refill This event counts for any cacheable read transaction returning data from the SCU for which the data source was outside the cluster.
0x002B	L3D_CACHE	Attributable level 3 data or unified cache access This event counts for any cacheable read, write or write-back transaction sent to the SCU.
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.

Event number	Mnemonic	Description
0x002F	L2D_TLB	Level 2 data TLB access Attributable level 2 unified TLB access. This event counts on any access to the L2 TLB (caused by a refill of any of the L1 TLBs). This event does not count if the MMU is disabled.
0x0031	REMOTE_ACCESS	Access to another socket in a multi-socket system This event counts any transactions returning data from another socket in a multi-socket system.
0x0034	DTLB_WALK	Data TLB access with at least one translation table walk Access to data TLB that caused a 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 Access to instruction TLB that caused a 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 This event counts any cacheable read transaction which returns a data source of 'interconnect cache', 'DRAM', 'remote' or 'inter-cluster peer'.
0x0037	LL_CACHE_MISS_RD	Last Level cache miss read This event counts any cacheable read transaction which returns a data source of 'DRAM', 'remote' or 'inter-cluster peer'.
0x0039	L1D_CACHE_LMISS_RD	Level 1 data cache long-latency read miss Level 1 data cache access, read. This event counts any load operation or translation table walk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_RD event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches. • Non-cacheable accesses.
0x003A	OP_RETIRED	Micro-operation architecturally executed This event 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 This event counts the number of operations executed by the core, including those that are executed speculatively and would not be executed in a simple sequential execution of the program.

Event number	Mnemonic	Description
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 core.
0x003D	STALL_SLOT_BACKEND	No operation sent for execution on a slot due to the backend 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 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 The counter counts on each Attributable cycle the number of instruction or operation slots that are not occupied by an instruction or operation Attributable to the PE.
0x0040	L1D_CACHE_RD	Level 1 data cache access, read Counts any load operation or translation table walk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_RD event causes this event to count. Cache maintenance instructions and prefetches are not counted. Non-cacheable accesses are not counted.
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. Cache maintenance instructions and prefetches are not counted. Non-cacheable accesses are not counted.
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) which hits in the L2 cache, L3 cache, or another core in the cluster.
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) which does not hit in the L2 cache, L3 cache, or another core in the cluster, and instead obtains data from outside the cluster.
0x0048	L1D_CACHE_INVAL	Level 1 data cache invalidate
0x0050	L2D_CACHE_RD	Level 2 data cache access, read This event counts any transaction issued from L1 caches which looks up in the L2 cache, including requests for instructions fetches and MMU table walks. The transaction is counted regardless of the source that generated it in the L1, being a load, store or prefetch request.
0x0051	L2D_CACHE_WR	Level 2 data cache access, write This event counts any full cache line write into the L2 cache which does not cause a linefill, including write-backs from L1 to L2, full-line writes which do not allocate into L1 and MMU descriptor hardware updates performed in L2.

Event number	Mnemonic	Description
0x0052	L2D_CACHE_REFILL_RD	Level 2 data cache refill, read This event counts any cacheable transaction generated by a read operation which causes data to be read from outside the L2. This event is counted on RXDAT, using the type of request to determine if the refill was for a read (line requested in any state).
0x0053	L2D_CACHE_REFILL_WR	Level 2 data cache refill, write This event counts any cacheable transaction generated by a store operation which causes data to be read from outside the L2. Counted on RXDAT, using the type of request to determine if the refill was for a write (line requested in unique state).
0x0056	L2D_CACHE_WB_VICTIM	Level 2 data cache write-back, victim This event counts any datafull write-back operation caused by allocations.
0x0057	L2D_CACHE_WB_CLEAN	Level 2 data cache write-back, cleaning, and coherency This event counts any datafull write-back operation caused by cache maintenance operations or external coherency requests.
0x0058	L2D_CACHE_INVAL	Level 2 data cache invalidate This event counts any cache maintenance operation which causes the invalidation of a line present in the L2 cache.
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 core and the SCU.
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 core and the SCU.
0x0066	MEM_ACCESS_RD	Data memory access, read This event counts memory accesses due to load instructions. The following instructions are not counted: • The following instructions are not counted: - Instruction fetches - Cache maintenance instructions. - Translation table walks. - Prefetches.
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.

Event number	Mnemonic	Description
0x006E	STREX_FAIL_SPEC	Exclusive operation speculatively executed, Store-Exclusive fail. Exclusive operation speculatively executed, STREX or STX fail.
0x006F	STREX_SPEC	Exclusive operation speculatively executed, Store-Exclusive. Exclusive operation speculatively executed, STREX or STX.
0x0070	LD_SPEC	Operation speculatively executed, load
0x0071	ST_SPEC	Operation speculatively executed, store
0x0073	DP_SPEC	Operation speculatively executed, integer data processing
0x0074	ASE_SPEC	Operation speculatively executed, Advanced SIMD
0x0075	VFP_SPEC	Operation speculatively executed, floating-point
0x0076	PC_WRITE_SPEC	Operation speculatively executed, software change of the PC
0x0077	CRYPTO_SPEC	Operation speculatively executed, Cryptographic instruction
0x007C	ISB_SPEC	Barrier speculatively executed, ISB
0x007D	DSB_SPEC	Barrier speculatively executed, DSB
0x007E	DMB_SPEC	Barrier speculatively executed, DMB
0x0081	EXC_UNDEF	Exception taken, other synchronous Counts the number of undefined exceptions taken locally
0x0082	EXC_SVC	Exception taken, Supervisor Call Exception taken locally, Supervisor Call
0x0083	EXC_PABORT	Exception taken, Instruction Abort Exception taken locally, Instruction Abort
0x0084	EXC_DABORT	Exception taken, Data Abort or SError Exception taken locally, Data Abort and SError
0x0086	EXC_IRQ	Exception taken, IRQ Exception taken locally, IRQ
0x0087	EXC_FIQ	Exception taken, FIQ Exception taken locally, FIQ
0x0088	EXC_SMC	Exception taken, Secure Monitor Call Exception taken locally, Secure Monitor Call
0x008A	EXC_HVC	Exception taken, Hypervisor Call Exception taken locally, Hypervisor Call
0x008B	EXC_TRAP_PABORT	Exception taken, Instruction Abort not Taken locally
0x008C	EXC_TRAP_DABORT	Exception taken, Data Abort or SError not Taken locally
0x008D	EXC_TRAP_OTHER	Exception taken, other traps not Taken locally
0x008E	EXC_TRAP_IRQ	Exception taken, IRQ not Taken locally
0x008F	EXC_TRAP_FIQ	Exception taken, FIQ not Taken locally

Event number	Mnemonic	Description
0x0090	RC_LD_SPEC	Release consistency operation speculatively executed, Load-Acquire
0x0091	RC_ST_SPEC	Release consistency operation speculatively executed, Store-Release
0x00A0	L3D_CACHE_RD	Level 3 cache read This event counts for any cacheable read transaction sent to the SCU.
0x4000	SAMPLE_POP	Statistical Profiling sample population The counter increments for each operation that might be sampled, whether or not the operation was sampled. Operations that are executed at an Exception level or Security state in which the Statistical Profiling Extension is disabled are not counted.
0x4001	SAMPLE_FEED	Statistical Profiling sample taken The counter increments each time the sample interval counter reaches zero and is reloaded, and the sample does not collide with the previous sample. Samples that are removed by filtering, or discarded, and not written to the Profiling Buffer are counted.
0x4002	SAMPLE_FILTRATE	Statistical Profiling sample taken and not removed by filtering The counter increments each time that a completed sample record is checked against the filters and not removed. Sample records that are not removed by filtering, but are discarded before being written to the Profiling Buffer because of a Profiling Buffer management event, are counted.
0x4003	SAMPLE_COLLISION	Statistical Profiling sample collided with previous sample The counter increments for each sample record that is taken when the previous sampled operation has not completed generating its sample record.
0x4004	CNT_CYCLES	Constant frequency cycles
0x4005	STALL_BACKEND_MEM	Memory stall cycles The counter is defined identically to STALL_BACKEND_MEM in the AMUv1 architecture.
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 Level 1 instruction cache.
0x4009	L2D_CACHE_LMISS_RD	Level 2 data cache long-latency read miss The counter counts each memory read access counted by L2D_CACHE that incurs additional latency because it returns data from outside the Level 2 data or unified cache of this PE.
0x400B	L3D_CACHE_LMISS_RD	Level 3 data cache long-latency read miss The counter counts each memory read access counted by L3D_CACHE that incurs additional latency because it returns data from outside the Level 3 data or unified cache of this PE.
0x400C	TRB_WRAP	Trace buffer current write pointer wrapped
0x400D	PMU_OVFS	PMU overflow, counters accessible to EL1 and EL0 **Note:** This event is exported to the trace unit, but cannot be counted in the PMU.

Event number	Mnemonic	Description
0x400E	TRB_TRIG	Trace buffer Trigger Event **Note:** This event is exported to the trace unit, but cannot be counted in the PMU.
0x400F	PMU_HOVFS	PMU overflow, counters reserved for use by EL2 **Note:** This event is exported to the trace unit, but cannot be counted in the PMU.
0x4010	TRCEXTOUT0	PE Trace Unit external output 0 **Note:** This event is not exported to the trace unit.
0x4011	TRCEXTOUT1	PE Trace Unit external output 1 **Note:** This event is not exported to the trace unit.
0x4012	TRCEXTOUT2	PE Trace Unit external output 2 **Note:** This event is not exported to the trace unit.
0x4013	TRCEXTOUT3	PE Trace Unit external output 3 **Note:** This event is not exported to the trace unit.
0x4018	CTI_TRIGOUT4	Cross Trigger Interface output trigger 4
0x4019	CTI_TRIGOUT5	Cross Trigger Interface output trigger 5
0x401A	CTI_TRIGOUT6	Cross Trigger Interface output trigger 6
0x401B	CTI_TRIGOUT7	Cross Trigger Interface output trigger 7
0x4020	LDST_ALIGN_LAT	Access with additional latency from alignment 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
0x4022	ST_ALIGN_LAT	Store with additional latency from alignment
0x4024	MEM_ACCESS_CHECKED	Checked data memory access
0x4025	MEM_ACCESS_RD_CHECKED	Checked data memory access, read
0x4026	MEM_ACCESS_WR_CHECKED	Checked data memory access, write
0x8005	ASE_INST_SPEC	Advanced SIMD operations speculatively executed
0x8006	SVE_INST_SPEC	SVE operation, including load/store
0x8014	FP_HP_SPEC	Half-precision floating-point operation speculatively executed
0x8018	FP_SP_SPEC	Single-precision floating-point operation speculatively executed
0x801C	FP_DP_SPEC	Double-precision floating-point operation speculatively executed
0x8074	SVE_PRED_SPEC	SVE predicated operations speculatively executed
0x8075	SVE_PRED_EMPTY_SPEC	SVE predicated operations with no active predicates speculatively executed
0x8076	SVE_PRED_FULL_SPEC	SVE predicated operations with all active predicates speculatively executed

Event number	Mnemonic	Description
0x8077	SVE_PRED_PARTIAL_SPEC	SVE predicated operations with partially active predicates speculatively executed
0x8079	SVE_PRED_NOT_FULL_SPEC	SVE predicated operations with no or partially active predicates speculatively executed
0x80BC	SVE_LDFF_SPEC	SVE First-fault load operations speculatively executed
0x80BD	SVE_LDFF_FAULT_SPEC	SVE First-fault load operations speculatively executed which set FFR bit to 0
0x80C0	FP_SCALE_OPS_SPEC	Scalable floating-point element operations speculatively executed
0x80C1	FP_FIXED_OPS_SPEC	Non-scalable floating-point element operations speculatively executed
0x80E3	ASE_SVE_INT8_SPEC	Advanced SIMD and SVE 8-bit integer operation speculatively executed
0x80E7	ASE_SVE_INT16_SPEC	Advanced SIMD and SVE 16-bit integer operation speculatively executed
0x80EB	ASE_SVE_INT32_SPEC	Advanced SIMD and SVE 32-bit integer operation speculatively executed
0x80EF	ASE_SVE_INT64_SPEC	Advanced SIMD and SVE 64-bit integer operation speculatively executed
0x8108	BR_IMMED_TAKEN_RETIRED	Instruction architecturally executed, immediate branch taken
0x810C	BR_INDNR_TAKEN_RETIRED	Instruction architecturally executed, indirect branch excluding procedure return retired
0x811D	BR_IND_RETIRED	Instruction architecturally executed, indirect branch
0x8128	DTLB_WALK_PERCYC	Total cycles, DTLB_WALK The counter counts by the number of data TLB walk events in progress on each processor cycle.
0x8129	ITLB_WALK_PERCYC	Total cycles, ITLB_WALK The counter counts by the number of instruction TLB walk events in progress on each processor cycle.
0x8136	DTLB_STEP	Data TLB translation table walk, step The counter counts each translation table walk access made by a refill of the data or unified TLB.
0x8137	ITLB_STEP	Instruction TLB translation table walk, step The counter counts each translation table walk access made by a refill of the instruction TLB.
0x8162	STALL_FRONTEND_FLUSH	No operation sent for execution due to the frontend flush recovery

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.

18.3 External register access permissions

The Cortex®-A715 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 **IMPLEMENTATION DEFINED** Performance Monitors registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table 18-2: Performance Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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
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

18.5 External PMU registers

The summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped PMU registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table 18-3: PMU registers summary

Offset	Name	Reset	Width	Description
0x600	PMPCSSR	—	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	—	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register

Offset	Name	Reset	Width	Description
0x60C	PMCID2SSR	—	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register
0x610	PMSSSR	—	32-bit	PMU Snapshot Status 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
0x6F0	PMSSCR	—	32-bit	PMU Snapshot Capture 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
0xE40	PMMIR	—	32-bit	Performance Monitors Machine Identification 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

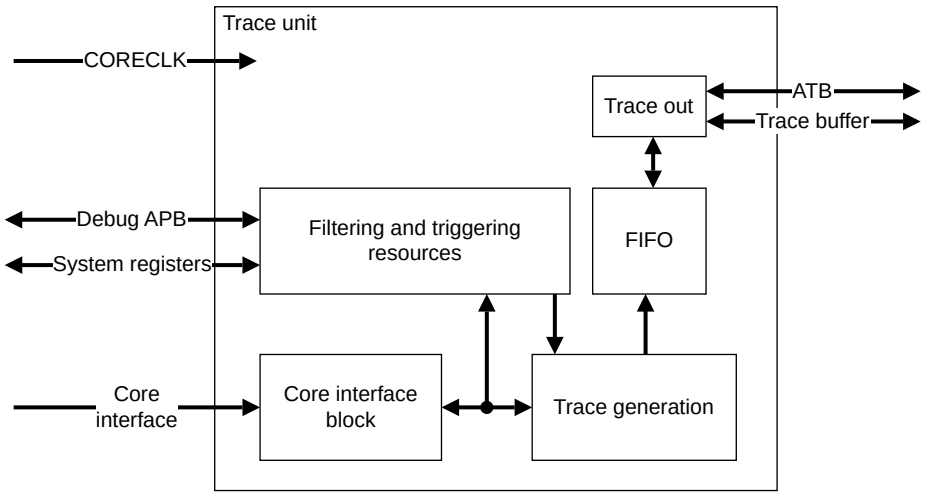
Offset	Name	Reset	Width	Description
0xFF4	PMCIDR1	—	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	—	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	—	32-bit	Performance Monitors Component Identification Register 3

19. Embedded Trace Extension support

The Cortex®-A715 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 Supplement Armv9, for Armv9-A architecture profile* 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 A715 core trace unit implements.

Table 19-1: Trace unit resources

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 Supplement Armv9, for Armv9-A architecture profile* for more information.

19.2 Trace unit generation options

The Cortex®-A715 core trace unit implements a set of generation options.

The following table shows the trace generation options, and indicates which of these options the Cortex®-A715 core trace unit implements.

Table 19-2: Trace unit generation options

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

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

19.3 Reset the trace unit

The reset for the trace 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

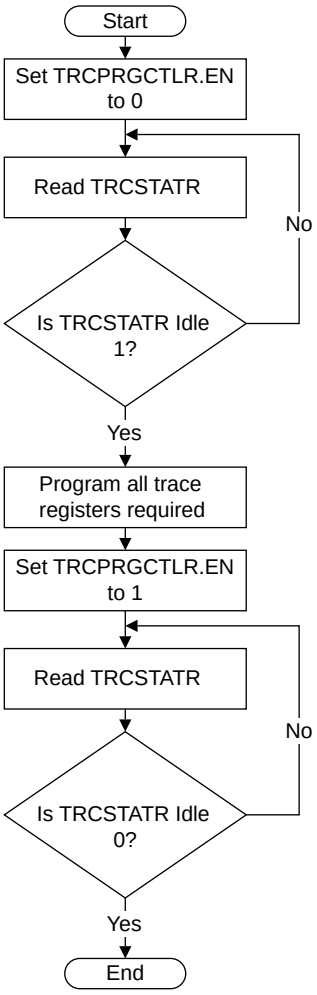
You program and read the trace unit registers using either the Debug APB interface or the System register interface.

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 Supplement Armv9, for Armv9-A architecture profile* for more information about the following registers:

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

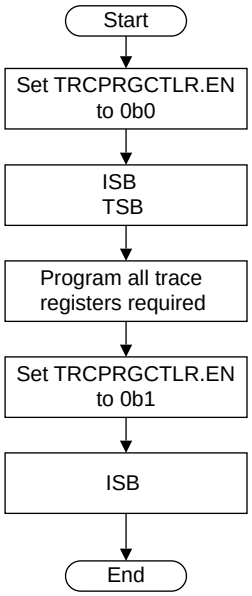
The following figure shows the flow for programming trace unit registers using the DebugBlock APB interface:

Figure 19-2: Programming trace unit registers using the DebugBlock 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®-A715 core supports an APB memory-mapped interface and a system register interface to trace unit registers.

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

19.6 Interaction with the Performance Monitoring Unit and Debug

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

Interaction with the PMU

The Cortex®-A715 core includes a PMU that enables events, such as cache misses and executed instructions, to be counted over time.

The PMU and trace unit function together.

Use of PMU events by the trace unit

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

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

Related information

18. Performance Monitors Extension support on page 98
18.1 Performance monitors events on page 98

19.7 ETE events

The Cortex®-A715 core trace unit collects events from other units in the design and uses numbers to reference these events.

Other than the events mentioned in 18.1 Performance monitors events on page 98, the events listed in the following table are also exported.

Table 19-3: ETE events

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

19.8 AArch64 Trace registers

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table 19-4: Trace registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCIDR8	2	1	C0	C0	6	–	64-bit	ID Register 8
TRCIMSPEC0	2	1	C0	C0	7	–	64-bit	IMP DEF Register 0
TRCIDR9	2	1	C0	C1	6	–	64-bit	ID Register 9
TRCIDR2	2	1	C0	C10	7	–	64-bit	ID Register 2
TRCIDR3	2	1	C0	C11	7	–	64-bit	ID Register 3
TRCIDR4	2	1	C0	C12	7	–	64-bit	ID Register 4
TRCIDR5	2	1	C0	C13	7	–	64-bit	ID Register 5
TRCIDR6	2	1	C0	C14	7	–	64-bit	ID Register 6
TRCIDR7	2	1	C0	C15	7	–	64-bit	ID Register 7

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCIDR10	2	1	C0	C2	6	–	64-bit	ID Register 10
TRCIDR11	2	1	C0	C3	6	–	64-bit	ID Register 11
TRCIDR12	2	1	C0	C4	6	–	64-bit	ID Register 12
TRCIDR13	2	1	C0	C5	6	–	64-bit	ID Register 13
TRCAUXCTLR	2	1	C0	C6	0	–	64-bit	Auxiliary Control Register
TRCIDR0	2	1	C0	C8	7	–	64-bit	ID Register 0
TRCIDR1	2	1	C0	C9	7	–	64-bit	ID Register 1
TRCDEVARCH	2	1	C7	C15	6	–	64-bit	Device Architecture Register
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

19.9 External ETE registers

The summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped ETE registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table 19-5: ETE registers summary

Offset	Name	Reset	Width	Description
0x018	TRCAUXCTLR	–	32-bit	Auxiliary Control Register
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
0xF00	TRCITCTRL	–	32-bit	Integration Mode Control Register
0xFA0	TRCCLAIMSET	–	32-bit	Claim Tag Set Register

Offset	Name	Reset	Width	Description
0xFA4	TRCCLAIMCLR	—	32-bit	Claim Tag Clear 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

20. Trace Buffer Extension support

The Cortex®-A715 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 Supplement Armv9, for Armv9-A architecture profile* for information on the TRBE register behaviors and access mechanisms.

20.2 Trace buffer register interface

The Cortex®-A715 core supports a System register interface to *TRace Buffer Extension* (TRBE) registers.

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

21. Activity Monitors Extension support

The Cortex®-A715 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®-A715 core implements seven counters in two groups, each of which is a 64-bit counter that counts a fixed event. Group 0 has four counters 0-3, and Group 1 has three counters 10-12.

21.1 Activity monitors access

The Cortex®-A715 core supports access to activity monitors from the System register interface and supports read-only memory-mapped access using the utility bus interface.

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

Access enable bit

The access enable bit AMUSERENR_EL0.EN controls access from EL0 to the activity monitors System registers.

The CPTR_EL2.TAM bit controls access from EL0 and EL1 to the activity monitors System registers. The CPTR_EL3.TAM bit controls access from EL0, EL1, and EL2 to the Activity Monitors Extension System registers. The AMUSERENR_EL0.EN bit is configurable at EL1, EL2, and EL3. All other controls, as well as the value of the counters, are configurable only at the highest implemented Exception level.

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

System register access

The activity monitors 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®-A715 core instance number in the DSU-110 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®-A715 core implements four activity monitors counters, 0-3, and three auxiliary counters, 10-12.

Each counter has the following characteristics:

- All events are counted in 64-bit wrapping counters that overflow when they wrap. There is no support for overflow status indication or interrupts.
- Any change in clock frequency, including when a `WFI` and `WFE` instruction stops the clock, can affect any counter.
- Events 0-3 and auxiliary events 10-12 are fixed, and the AMEVTYPER0<n>_EL0 and AMEVTYPER1<n>_EL0 evtCount bits are read-only.
- The activity monitor counters are reset to zero on a Cold reset of the power domain of the core. When the core is not in reset, activity monitoring is available.

21.3 Activity monitors events

Activity monitors events in the Cortex®-A715 core are either fixed or programmable, and they map to the activity monitors counters.

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

Table 21-1: Mapping of counters to fixed events

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR00	CPU_CYCLES	`0x0011`	Core frequency cycles
AMEVCNTR01	CNT_CYCLES	`0x4004`	Constant frequency cycles
AMEVCNTR02	Instructions retired	`0x0008`	Instruction architecturally executed This counter increments for every instruction that is executed architecturally, including instructions that fail their condition code check.

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR03	STALL_BACKEND_MEM	0x4005	Memory stall cycles This counter counts cycles in which the core is unable to dispatch instructions from the front end to the back end due to a back end stall caused by a miss in the last level of cache within the core clock domain.
AMEVCNTR10	MPMM_THRESHOLD_GEAR0	0x0300	*Maximum Power Mitigation System* (MPMM) Gear 0 activity period threshold exceeded
AMEVCNTR11	MPMM_THRESHOLD_GEAR1	0x0301	*Maximum Power Mitigation System* (MPMM) Gear 1 activity period threshold exceeded
AMEVCNTR12	MPMM_THRESHOLD_GEAR2	0x0302	*Maximum Power Mitigation System* (MPMM) Gear 2 activity period threshold exceeded

21.4 AArch64 Activity Monitors registers

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table 21-2: Activity Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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
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
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

21.5 External AMU registers

The summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped AMU registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table 21-3: AMU registers summary

Offset	Name	Reset	Width	Description
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
0xCE0	AMCGCR	—	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	—	32-bit	Activity Monitors Configuration Register
0xE08	AMIIDR	—	32-bit	Activity Monitors Implementation Identification Register
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

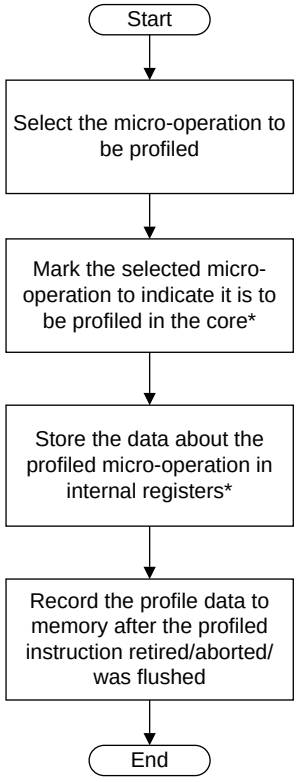
22. Statistical Profiling Extension Support

The Cortex®-A715 core implements the optional *Statistical Profiling Extension* (SPE) to the Arm®v8.2-A architecture. The SPE provides a statistical view of the performance characteristics of executed instructions that software writers can use to optimize their code for better performance.

The Cortex®-A715 core profiles micro-operations to minimize the amount of logic necessary to support the SPE.

The following figure shows the SPE behavior in the Cortex®-A715 core.

Figure 22-1: SPE behavior



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

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

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

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

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

22.1 Statistical Profiling Extension events packet

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

The following table shows the events defined in the 32-bit events packet implemented in the Cortex®-A715 core.

Table 22-1: SPE events packet

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

22.2 Statistical Profiling Extension data source packet

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

The following table shows the data source defined in the 8-bit data source packet implemented in the Cortex®-A715 core.

Table 22-2: SPE data source packet

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

Appendix A AArch64 registers

This appendix contains the descriptions for the Cortex®-A715 core 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 **IMPLEMENTATION DEFINED** Generic System Control registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table A-1: Generic System Control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AIDR_EL1	3	1	C0	C0	7	–	64-bit	Auxiliary ID Register
ACTLR_EL1	3	0	C1	C0	1	–	64-bit	Auxiliary Control Register (EL1)
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
ACTLR_EL3	3	6	C1	C0	1	–	64-bit	Auxiliary Control Register (EL3)
AMAIR_EL2	3	0	C10	C3	0	–	64-bit	Auxiliary Memory Attribute Indirection Register (EL2)
LORID_EL1	3	0	C10	C4	7	–	64-bit	LORegionID (EL1)
AMAIR_EL1	3	5	C10	C3	0	–	64-bit	Auxiliary Memory Attribute Indirection Register (EL1)
AMAIR_EL3	3	6	C10	C3	0	–	64-bit	Auxiliary Memory Attribute Indirection Register (EL3)
IMP_CPUCFR_EL1	3	0	C15	C0	0	–	64-bit	CPU Configuration Register
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
IMP_CPUACTLR3_EL1	3	0	C15	C1	2	–	64-bit	CPU Auxiliary Control Register
IMP_CPUACTLR4_EL1	3	0	C15	C1	3	–	64-bit	CPU Auxiliary Control Register
IMP_CPUECTLR_EL1	3	0	C15	C1	4	–	64-bit	CPU Extended Control Register
IMP_CPUECTLR2_EL1	3	0	C15	C1	5	–	64-bit	CPU Extended Control Register
IMP_CPUPWRCTLR_EL1	3	0	C15	C2	7	–	64-bit	CPU Power Control Register
IMP_CLUSTERACTLR_EL1	3	0	C15	C3	3	–	64-bit	Cluster Auxiliary Control Register
IMP_ATCR_EL2	3	0	C15	C7	0	–	64-bit	CPU Auxiliary Translation Control Register
IMP_AVTCR_EL2	3	4	C15	C7	1	–	64-bit	CPU Auxiliary Virtualization Translation Control Register
IMP_ATCR_EL1	3	5	C15	C7	0	–	64-bit	CPU Auxiliary Translation Control Register
IMP_ISIDE_DATA0_EL3	3	6	C15	C0	0	–	64-bit	RAMINDEX Instruction Data register 0
IMP_ISIDE_DATA1_EL3	3	6	C15	C0	1	–	64-bit	RAMINDEX Instruction Data register 1
IMP_ISIDE_DATA2_EL3	3	6	C15	C0	2	–	64-bit	RAMINDEX Instruction Data register 2
IMP_MMU_DATA0_EL3	3	6	C15	C0	3	–	64-bit	RAMINDEX TLB Data register 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_MMU_DATA1_EL3	3	6	C15	C0	4	–	64-bit	RAMINDEX TLB Data register 1
IMP_MMU_DATA2_EL3	3	6	C15	C0	5	–	64-bit	RAMINDEX TLB Data register 2
IMP_DSIDE_DATA0_EL3	3	6	C15	C1	0	–	64-bit	RAMINDEX L1D Data register 0
IMP_DSIDE_DATA1_EL3	3	6	C15	C1	1	–	64-bit	RAMINDEX L1D Data register 1
IMP_DSIDE_DATA2_EL3	3	6	C15	C1	2	–	64-bit	RAMINDEX L1D Data register 2
IMP_L2_DATA0_EL3	3	6	C15	C1	3	–	64-bit	RAMINDEX L2 Data register 0
IMP_L2_DATA2_EL3	3	6	C15	C1	4	–	64-bit	RAMINDEX L2 Data register 2
IMP_L2_DATA1_EL3	3	6	C15	C1	5	–	64-bit	RAMINDEX L2 Data register 1
IMP_CLUSTERCDBG_EL3	3	6	C15	C4	7	–	64-bit	Cluster Cache Debug Register
IMP_ATCR_EL3	3	6	C15	C7	0	–	64-bit	CPU Auxiliary Translation Control Register
IMP_CPUPSELR_EL3	3	6	C15	C8	0	–	64-bit	Selected Instruction Private Control Register
IMP_CPUPCR_EL3	3	6	C15	C8	1	–	64-bit	Selected Instruction Private Control Register
IMP_CPUPOR_EL3	3	6	C15	C8	2	–	64-bit	Selected Instruction Patch Opcode Register
IMP_CPUPMR_EL3	3	6	C15	C8	3	–	64-bit	Selected Instruction Patch Mask Register
IMP_CPUPOR2_EL3	3	6	C15	C8	4	–	64-bit	Selected Instruction Patch Opcode Register 2
IMP_CPUPMR2_EL3	3	6	C15	C8	5	–	64-bit	Selected Instruction Private Mask Register 2
IMP_CPUPFR_EL3	3	6	C15	C8	6	–	64-bit	Selected Instruction Private Flag Register
FPCR	3	3	C4	C4	0	–	64-bit	Floating-point Control Register
AFSR0_EL2	3	0	C5	C1	0	–	64-bit	Auxiliary Fault Status Register 0 (EL2)
AFSR1_EL2	3	0	C5	C1	1	–	64-bit	Auxiliary Fault Status Register 1 (EL2)
AFSR0_EL1	3	5	C5	C1	0	–	64-bit	Auxiliary Fault Status Register 0 (EL1)
AFSR1_EL1	3	5	C5	C1	1	–	64-bit	Auxiliary Fault Status Register 1 (EL1)
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)

A.1.1 AIDR_EL1, Auxiliary ID Register

Provides **IMPLEMENTATION DEFINED** identification information.

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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_aidr_el1 bit assignments

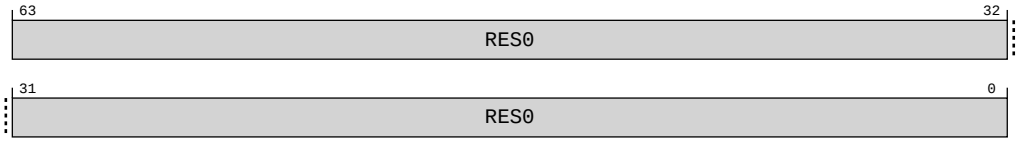


Table A-2: AIDR_EL1 bit descriptions

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

A.1.2 ACTLR_EL1, Auxiliary Control Register (EL1)

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

Note

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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_actlr_el1 bit assignments

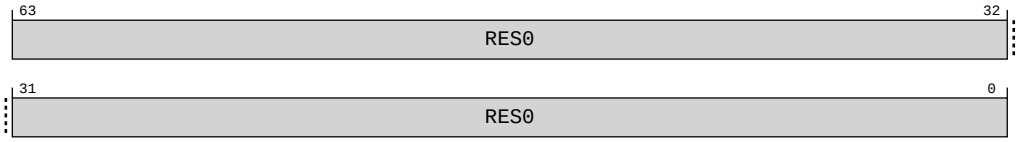


Table A-4: ACTLR_EL1 bit descriptions

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.1.3 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-3: AArch64_actlr_el2 bit assignments

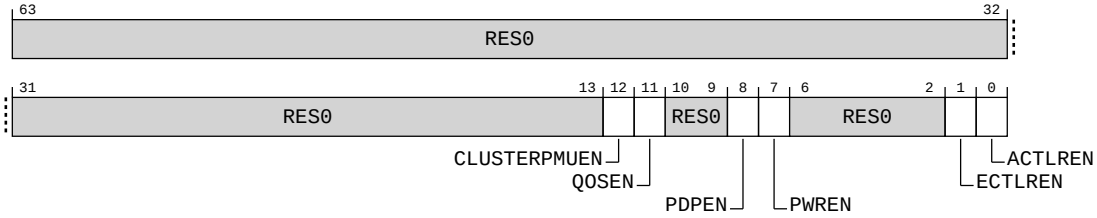


Table A-7: ACTLR_EL2 bit descriptions

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

Bits	Name	Description	Reset
[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
[10:9]	**RES0**	Reserved	**RES0**
[8]	PDPEN	Perfomance defined power enable. Possible values of this bit are: **0b0** This control causes writes to AArch64-IMP_CPUPPMPDPCR_EL1 at EL1 to be trapped. **0b1** This control does not cause any instructions to be trapped.	x
[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**
[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.1.4 HACR_EL2, Hypervisor Auxiliary Control Register

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


Note

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

Configurations

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

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

Attributes

Width

64

Functional group

Generic System Control

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-4: AArch64_hacr_el2 bit assignments

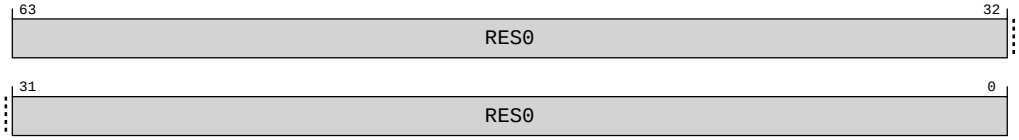


Table A-10: HACR_EL2 bit descriptions

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.1.5 ACTLR_EL3, Auxiliary Control Register (EL3)

Provides **IMPLEMENTATION DEFINED** configuration and control options for EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-5: AArch64_actlr_el3 bit assignments

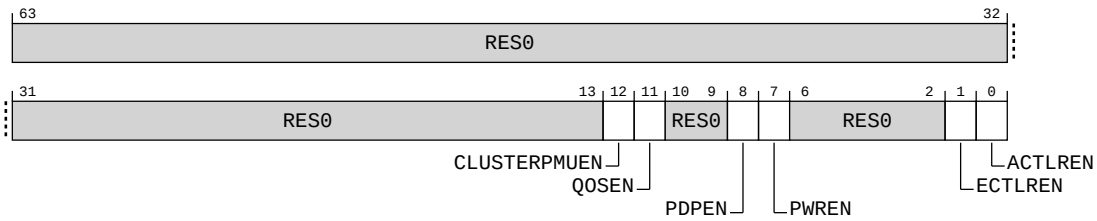


Table A-13: ACTLR_EL3 bit descriptions

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
[10:9]	**RES0**	Reserved	**RES0**
[8]	PDPEN	Perfomance defined power enable. Possible values of this bit are: **0b0** This control causes writes to AArch64-IMP_CPUPPMPDPCR_EL1 at EL1 to be trapped. **0b1** This control does not cause any instructions to be trapped.	x
[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**
[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_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.1.6 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by AArch64-MAIR_EL2.

Configurations

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

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

Attributes

Width

64

Functional group

Generic System Control

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-6: AArch64_amair_el2 bit assignments

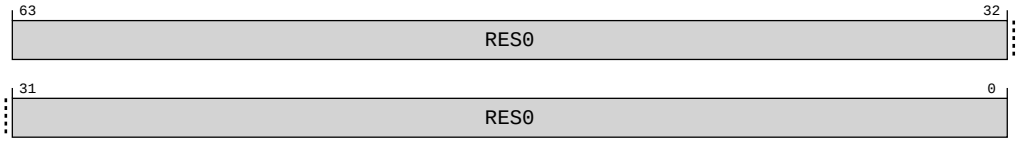


Table A-16: AMAIR_EL2 bit descriptions

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);
    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);
    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.1.7 LORID_EL1, LORegionID (EL1)

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

Configurations

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

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100 xxxx xxxx 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-7: AArch64_lorid_el1 bit assignments

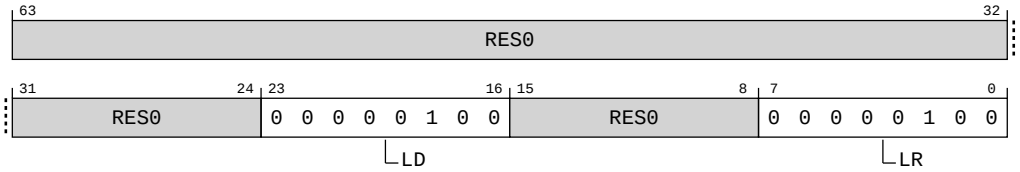


Table A-21: LORID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	**RES0**	Reserved	**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	0x04
[15:8]	**RES0**	Reserved	**RES0**
[7:0]	LR	Number of LORegions supported by the PE. This is an 8-bit binary number. **Note:** If LORID_EL1 indicates that no LORegions are implemented, then LoadLOAcquire and StoreLORelease will behave as LoadAcquire and StoreRelease. **0b00000100** Four LORegions are supported	0x04

Access

MRS <Xt>, LORID_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b111

Accessibility

MRS <Xt>, LORID_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TLOR == '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.1.8 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register (EL1)

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-8: AArch64_amair_el1 bit assignments

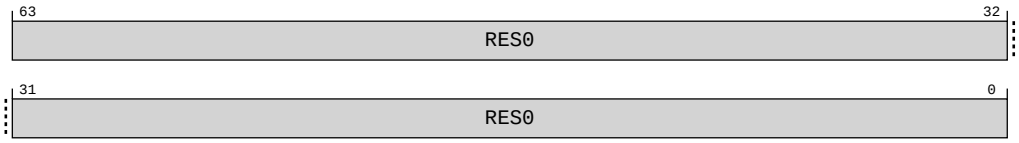


Table A-23: AMAIR_EL1 bit descriptions

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);
    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);
    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.1.9 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register (EL3)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by AArch64-MAIR_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-9: AArch64_amair_el3 bit assignments

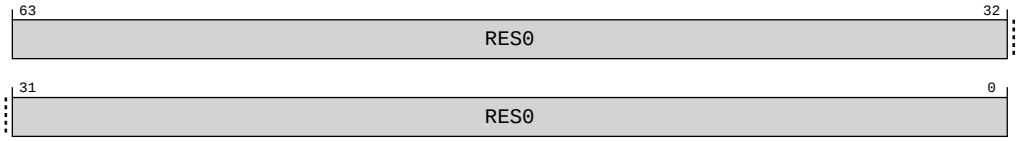


Table A-28: AMAIR_EL3 bit descriptions

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.1.10 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

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_cpucfr_el1 bit assignments

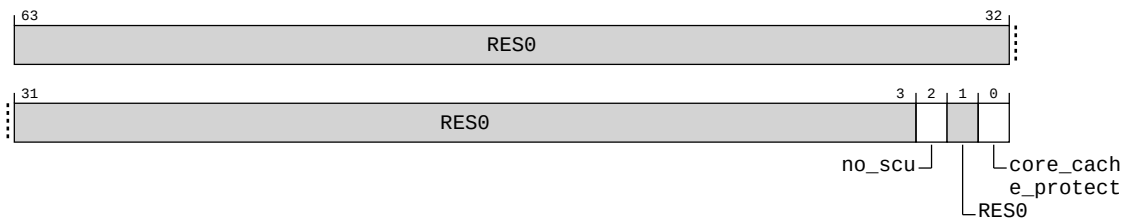


Table A-31: IMP_CPUCFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:3]	**RES0**	Reserved	**RES0**
[2]	no_scu	Indicates whether the SCU is present or not. Possible values of this bit are: **0b0** The SCU is present. **0b1** The SCU is not present.	x
[1]	**RES0**	Reserved	**RES0**
[0]	core_cache_protect	Indicates whether ECC is present or not. Possible values of this field are: **0b0** ECC is not present. **0b1** ECC is present.	x

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.1.11 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-11: AArch64_imp_cpuactlr_el1 bit assignments

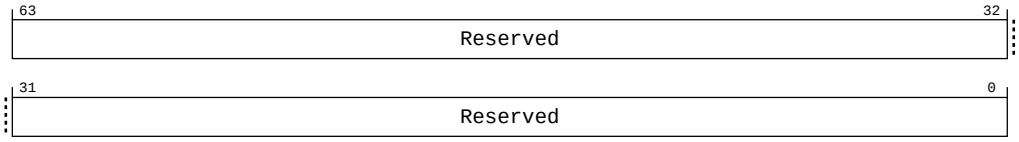


Table A-33: IMP_CPUACTLR_EL1 bit descriptions

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

Access

MSR S3_0_C15_C1_0, <Xt>

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

MRS <Xt>, S3_0_C15_C1_0

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

Accessibility

MSR S3_0_C15_C1_0, <Xt>

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

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

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;
```

A.1.12 IMP_CPUACTLR2_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-12: AArch64_imp_cpuactlr2_el1 bit assignments

Table A-36: IMP_CPUACTLR2_EL1 bit descriptions

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

Access

MSR S3_0_C15_C1_1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b001

MRS <Xt>, S3_0_C15_C1_1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b001

Accessibility

MSR S3_0_C15_C1_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ACTLREN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if 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];
```

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;
```

A.1.13 IMP_CPUACTLR3_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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-13: AArch64_imp_cpuactlr3_el1 bit assignments

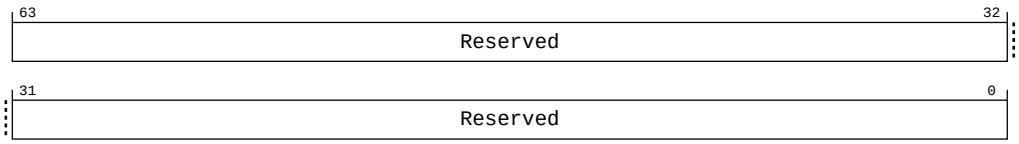


Table A-39: IMP_CPUACTLR3_EL1 bit descriptions

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

Access

MSR S3_0_C15_C1_2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b010

MRS <Xt>, S3_0_C15_C1_2

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b010

Accessibility

MSR S3_0_C15_C1_2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ACTLREN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR3_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if 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];
```

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;
```

A.1.14 IMP_CPUACTLR4_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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-14: AArch64_imp_cpuactlr4_el1 bit assignments

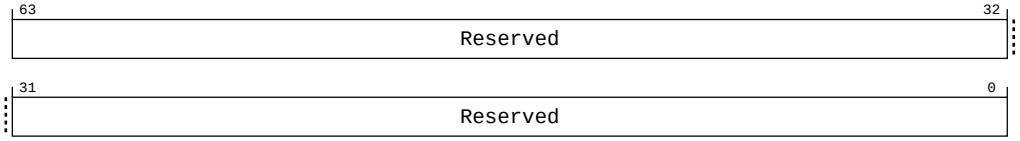


Table A-42: IMP_CPUACTLR4_EL1 bit descriptions

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

Access

MSR S3_0_C15_C1_3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b011

MRS <Xt>, S3_0_C15_C1_3

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b011

Accessibility

MSR S3_0_C15_C1_3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ACTLREN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR4_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR4_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR4_EL1 = X[t];
```

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_CPUACTLR4_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUACTLR4_EL1;
elsif PSTATE.EL == EL3 then
```

```
    return IMP_CPUACTLR4_EL1;
```

A.1.15 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 0110 0000 0000 0000 0111 0010 0000 0110 0010 0000 1011 xxxx 0001 010x

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-15: AArch64_imp_cpuectlr_el1 bit assignments

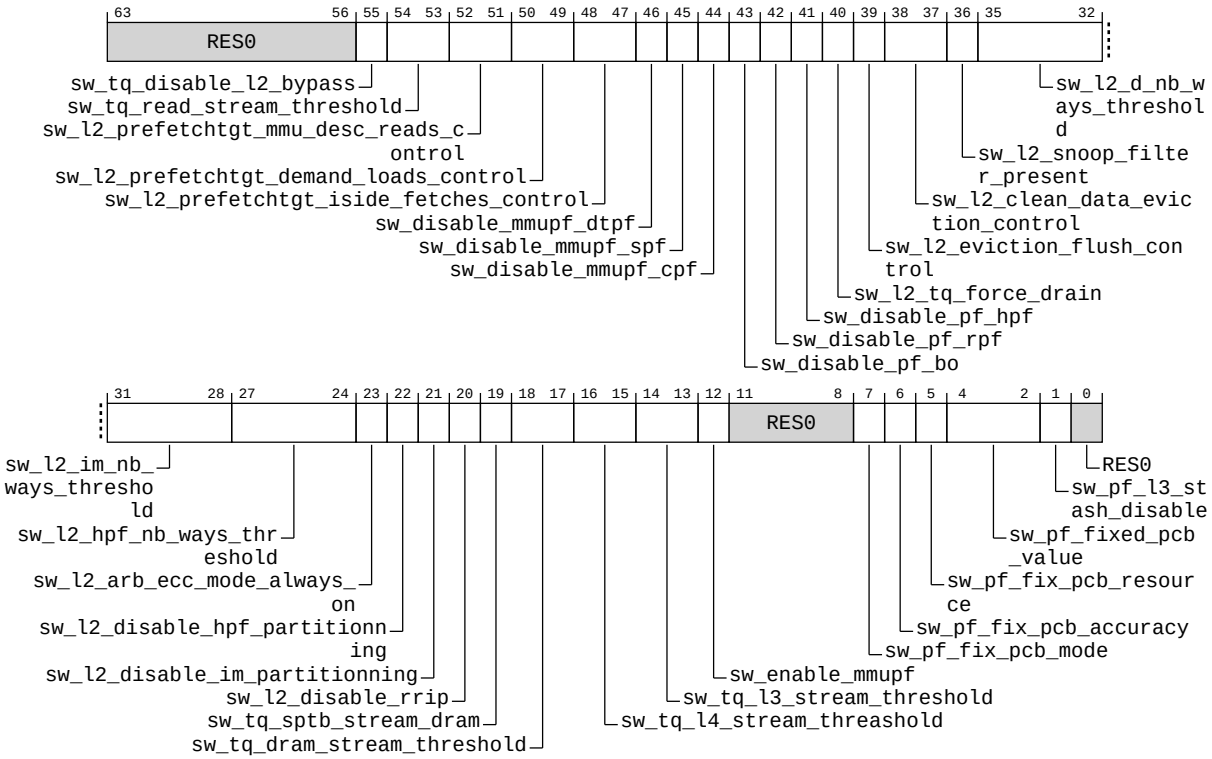


Table A-45: IMP_CPUECTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[55]	sw_tq_disable_l2_bypass	Disable the heuristics that sends L1 evictions directly to L3, bypassing the L2 **0b0** Enabled **0b1** Disabled	0b0
[54:53]	sw_tq_read_stream_threshold	Configure thresholds for transforming WriteEvictOrEvict into Evict **0b00** 256 KB **0b01** 512 KB **0b10** 1024 KB **0b11** Disable	0b11

Bits	Name	Description	Reset
[52:51]	sw_l2_prefetchtgt_mmu_desc_reads_control	Control activation of the PrefetchTgt for MMU descriptors reads **0b00** PrefetchTgt are not sent for MMU descriptors reads **0b01** Conservatively generate PrefetchTgt for cacheable requests from the MMU, always generate for noncacheable **0b10** Aggressively generate PrefetchTgt for cacheable requests from the MMU, always generate for noncacheable **0b11** Always generate PrefetchTgt for cacheable requests from the MMU, always generate for noncacheable	0b00
[50:49]	sw_l2_prefetchtgt_demand_loads_control	Control activation of the PrefetchTgt for the Demand Loads/Stores **0b00** PrefetchTgt are not sent for Demand Loads/Stores **0b01** Conservatively generate PrefetchTgt for cacheable requests for Demand Loads/Stores, always generate for noncacheable **0b10** Aggressively generate PrefetchTgt for cacheable requests for Demand Loads/Stores, always generate for noncacheable **0b11** Always generate PrefetchTgt for cacheable requests for Demand Loads/Stores, always generate for noncacheable	0b00
[48:47]	sw_l2_prefetchtgt_iside_fetches_control	Control activation of the PrefetchTgt for the Iside fetches **0b00** PrefetchTgt are not sent for Iside fetches **0b01** Conservatively generate PrefetchTgt for cacheable requests from the Iside, always generate for noncacheable **0b10** Aggressively generate PrefetchTgt for cacheable requests from the Iside, always generate for noncacheable **0b11** Always generate PrefetchTgt for cacheable requests from the Iside, always generate for noncacheable	0b00
[46]	sw_disable_mmupf_dtpf	Control activation of MMU debug & trace prefetcher (DTPF) **0b0** MMU D&T prefetcher is enabled allowing the prefetching of next/previous page for each D&T request (requires feature bit ENABLE_MMUPF in CPUECTLR to be set as well) **0b1** MMU D&T prefetcher is disabled	0b0

Bits	Name	Description	Reset
[45]	sw_disable_mmupf_spf	Control activation of MMU stream prefetcher (SPF) **0b0** MMU stream prefetcher is enabled allowing the prefetching of next/previous page when a stream is detected (requires feature bit ENABLE_MMUPF in CPUECTLR to be set as well) **0b1** MMU stream prefetcher is disabled	0b0
[44]	sw_disable_mmupf_cpf	Control activation of MMU coalescing prefetcher (CPF) **0b0** MMU coalescing prefetcher is enabled allowing the prefetching of next/previous clusters in case of coalesced entries (requires feature bit ENABLE_MMUPF in CPUECTLR to be set as well) **0b1** MMU coalescing prefetcher is disabled	0b0
[43]	sw_disable_pf_bo	Disable Best Offset Prefetcher **0b0** Enabled **0b1** Disabled	0b0
[42]	sw_disable_pf_rpf	Disable Region Prefetcher **0b0** Enabled **0b1** Disabled	0b0
[41]	sw_disable_pf_hpf	Disable History Prefetcher **0b0** Enabled **0b1** Disabled	0b0
[40]	sw_l2_tq_force_drain	Force all evictions (L1/L2) to be drained out of TQ **0b0** Evictions are drained normally **0b1** Evictions are forced to drain ASAP	0b0

Bits	Name	Description	Reset
[39]	sw_l2_eviction_flush_control	Controls whether hardware cache flushes and DC CISW instructions send data when evicting clean cache-lines on the CHI interface **0b0** Disables sending data when hardware cache flushes or DC CISW instructions evict a clean cache-line. Sending of Evict transactions is controlled by Downstream Snoop Filter Present. This is the reset value **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 Downstream Snoop Filter Present	0b0
[38:37]	sw_l2_clean_data_eviction_control	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.	0b11
[36]	sw_l2_snoop_filter_present	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
[35:32]	sw_l2_d_nb_ways_threshold	D-side ways partition in L2 cache. The value specified indicates the number of ways dedicated to requests from L1 data cache. It is expected within the range 0b0000-0b1000, an higher value will be ignored and set to 0b1000. The specified way partitioning is not guaranteed if the sum with sw_l2_im_nb_ways_threshold and sw_l2_hpf_nb_ways_threshold is not equal to 8, the number of L2 cache ways. If the sum is lower than 8, the remaining ways will be used for D and I/MMU requests. Reset value is 0b0010 and is the recommended value for this product	0b0010
[31:28]	sw_l2_im_nb_ways_threshold	I-side/MMU ways partition in L2 cache. The value specified indicates the number of ways dedicated to requests from L1 I-cache or MMU. It is expected within the range 0b0000-0b1000, an higher value will be ignored and set to 0b1000. The specified way partitioning is not guaranteed if the sum with sw_l2_d_nb_ways_threshold and sw_l2_hpf_nb_ways_threshold is not equal to 8, the number of L2 cache ways. If the sum is lower than 8, the remaining ways will be used for D and I/MMU requests. Reset value is 0b0000 and is the recommended value for this product	0b0000

Bits	Name	Description	Reset
[27:24]	sw_l2_hpf_nb_ways_threshold	HPF ways partition in L2 cache. The value specified indicates the number of ways dedicated to requests from HPF. It is expected within the range `0b0000-0b1000`, an higher value will be ignored and set to `0b1000`. The specified way partitioning is not guaranteed if the sum with sw_l2_d_nb_ways_threshold and sw_l2_im_nb_ways_threshold is not equal to 8, the number of L2 cache ways. If the sum is lower than 8, the remaining ways will be used for D and I/MMU requests. Reset value is `0b0110` and is the recommended value for this product	`0b0110`
[23]	sw_l2_arb_ecc_mode_always_on	Force one more cycle in the L2 pipeline to allow for inline ECC error corrections on the Tag RAM **0b0** Default **0b1** Force	`0b0`
[22]	sw_l2_disable_hpf_partitionning	Disable L2 cache replacement policy partitioning for History Prefetcher **0b0** Default **0b1** Disable	`0b0`
[21]	sw_l2_disable_im_partitionning	Disable L2 cache replacement policy partitioning for L1I and MMU **0b0** Default **0b1** Disable	`0b1`
[20]	sw_l2_disable_rrip	Disable RRIP replacement policy in L2 cache, falling back to random policy **0b0** Default **0b1** Disable	`0b0`
[19]	sw_tq_sptb_stream_dram	Force all writes for Trace and Statistical Profiling buffers to external memory **0b0** Default **0b1** Disable	`0b0`
[18:17]	sw_tq_dram_stream_threshold	Configure thresholds for streaming writes to external memory **0b00** 1024 KB **0b01** 2048 KB **0b10** 4096 KB **0b11** Disable	`0b00`

Bits	Name	Description	Reset
[16:15]	sw_tq_l4_stream_threashold	Configure thresholds for streaming writes to L4 cache **0b00** 256 KB **0b01** 512 KB **0b10** 1024 KB **0b11** Disable	0b01
[14:13]	sw_tq_l3_stream_threshold	Configure thresholds for streaming writes to L3 cache **0b00** 16 KB **0b01** 32 KB **0b10** 64 KB **0b11** Disable	0b01
[12]	sw_enable_mmupf	Control activation of MMU prefetchers **0b0** MMU prefetchers are disabled **0b1** MMU prefetchers are enabled and can send translation requests to MMU pipeline	0b1
[11:8]	**RES0**	Reserved	**RES0**
[7]	sw_pf_fix_pcb_mode	Disable L2 PF dynamic modes and fix to a static value given by sw_pf_fixed_pcb_value **0b0** Dynamic behaviour is enabled **0b1** Modes are fixed to a static value	0b0
[6]	sw_pf_fix_pcb_accuracy	Disable L2 PF dynamic accuracy feedback and fix to a static value given by sw_pf_fixed_pcb_value **0b0** Dynamic behaviour is enabled **0b1** Accuracy feedback is fixed to a static value	0b0
[5]	sw_pf_fix_pcb_resource	Disable L2 PF dynamic resource feedback and fix to a static value given by sw_pf_fixed_pcb_value **0b0** Dynamic behaviour is enabled **0b1** Resource feedback is fixed to a static value	0b0

Bits	Name	Description	Reset
[4:2]	sw_pf_fixed_pcb_value	Static value for L2 PF mode, accuracy feedback or resource feedback if dynamic behaviour is disabled **0b000** Mode = 0 (turn PF off), accuracy_feedback = VERY_LOW, resource_feedback = FULLY_SATURATED **0b001** Mode = 1, accuracy_feedback = LOW, resource_feedback = PARTLY_SATURATED **0b010** Mode = 2, accuracy_feedback = MEDIUM, resource_feedback = NORMAL **0b011** Mode = 3, accuracy_feedback = HIGH, resource_feedback = LOW **0b100** Mode = 4, accuracy_feedback = VERY_HIGH, resource_feedback = EMPTY **0b101** Mode = 5 (most aggressive), invalid for accuracy and resource feedback **0b110** Invalid **0b111** Invalid	0b101
[1]	sw_pf_l3_stash_disable	Prevents any PF to send stash requests on CHI **0b0** Enabled **0b1** Disabled	0b0
[0]	**RES0**	Reserved	**RES0**

Access

MSR S3_0_C15_C1_4, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b100

MRS <Xt>, S3_0_C15_C1_4

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b100

Accessibility

MSR S3_0_C15_C1_4, <Xt>

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

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;
```

A.1.16 IMP_CPUECTLR2_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 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 0000 0001

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-16: AArch64_imp_cpuectlr2_el1 bit assignments

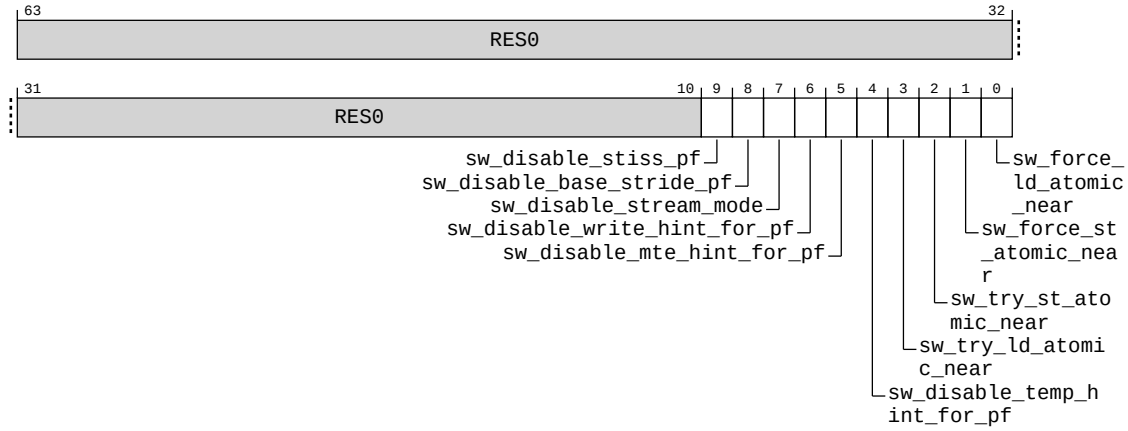


Table A-48: IMP_CPUECTLR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:10]	**RES0**	Reserved	**RES0**
[9]	sw_disable_stiss_pf	Disable Store Issue Prefetcher **0b0** Store Issue Prefetcher is enabled **0b1** Store Issue Prefetcher is disabled	0b0
[8]	sw_disable_base_stride_pf	Disable Stride Prefetcher **0b0** Stride PF is enabled **0b1** Stride PF is disabled	0b0

Bits	Name	Description	Reset
[7]	sw_disable_stream_mode	Disable stream of writes to L2 and beyond **0b0** Stream mode is enabled **0b1** Stream mode is disabled	0b0
[6]	sw_disable_write_hint_for_pf	Disable write access hint for prefetch. **0b0** Write access hint is enabled **0b1** Write access hint is disabled	0b0
[5]	sw_disable_mte_hint_for_pf	Disable mte access hint for prefetch. **0b0** MTE access hint is enabled **0b1** MTE access hint is disabled	0b0
[4]	sw_disable_temp_hint_for_pf	Disable temp access hint for prefetch. **0b0** Temp access hint is enabled **0b1** Temp access hint is disabled	0b0
[3]	sw_try_ld_atomic_near	Try atomic load as near even if misses in cache. If the line is evicted, move to far **0b0** Don't try to execute atomic load in near mode if miss **0b1** Try to execute atomic load in near mode if miss	0b0
[2]	sw_try_st_atomic_near	Try atomic store as near even if misses in cache. If the line is evicted, move to far **0b0** Don't try to execute atomic store in near mode if miss **0b1** Try to execute atomic store in near mode if miss	0b0
[1]	sw_force_st_atomic_near	Always execute atomic store in near mode **0b0** Don't force atomic store to execute in near mode **0b1** Force atomic store to execute in near mode	0b0
[0]	sw_force_ld_atomic_near	Always execute atomic load in near mode **0b0** Don't force atomic load to execute in near mode **0b1** Force atomic load to execute in near mode	0b1

Access

MSR S3_0_C15_C1_5, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b101

MRS <Xt>, S3_0_C15_C1_5

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0001	0b101

Accessibility

MSR S3_0_C15_C1_5, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && ACTLR_EL3.ECTLREN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ECTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR2_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ECTLREN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR2_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUECTLR2_EL1 = X[t];
```

MRS <Xt>, S3_0_C15_C1_5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUECTLR2_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUECTLR2_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUECTLR2_EL1;
```

A.1.17 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-17: AArch64_imp_cpupwrctlr_el1 bit assignments

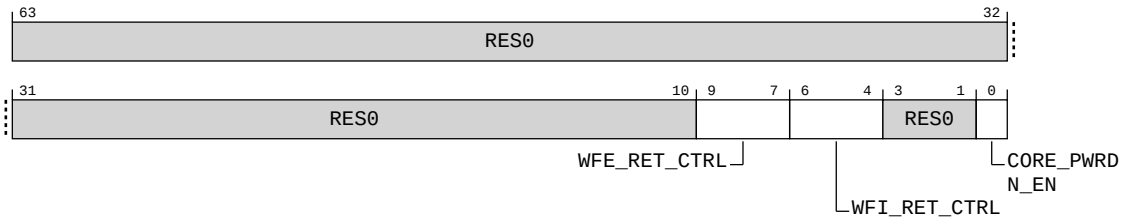


Table A-51: IMP_CPUPWRCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:10]	RES0	Reserved	RES0

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

MSR S3_0_C15_C2_7, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b111

MRS <Xt>, S3_0_C15_C2_7

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b111

Accessibility

MSR S3_0_C15_C2_7, <Xt>

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

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;
```

A.1.18 IMP_CLUSTERACTLR_EL1, Cluster Auxiliary Control Register

These register bits are reserved for Arm test purposes only and must not be used except under direction from Arm.

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-18: AArch64_imp_clusteractlr_el1 bit assignments

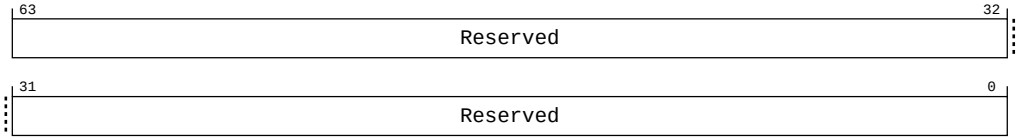


Table A-54: IMP_CLUSTERACTLR_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C3_3

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0011	0b011

MSR S3_0_C15_C3_3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0011	0b011

Accessibility

MRS <Xt>, S3_0_C15_C3_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_CLUSTERACTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CLUSTERACTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CLUSTERACTLR_EL1;
```

MSR S3_0_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);
    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_CLUSTERACTLR_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_CLUSTERACTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CLUSTERACTLR_EL1 = X[t];
```

A.1.19 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

When AArch64-HCR_EL2.E2H == 1

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

When AArch64-HCR_EL2.E2H == 0

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When AArch64-HCR_EL2.E2H == 1

Figure A-19: AArch64_imp_atcr_el2 bit assignments

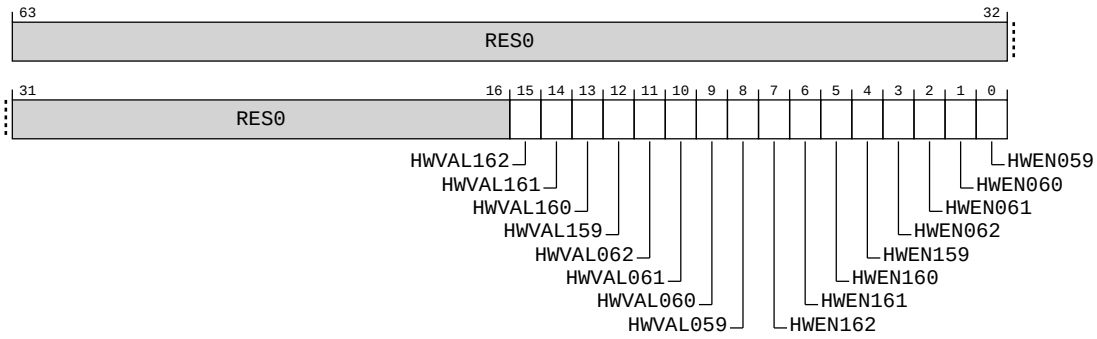


Table A-57: IMP_ATCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN162 is set.	0b0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN161 is set.	0b0
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN160 is set.	0b0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN159 is set.	0b0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN062 is set.	0b0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN061 is set.	0b0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN060 is set.	0b0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN059 is set.	0b0

Bits	Name	Description	Reset
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0

When AArch64-HCR_EL2.E2H == 0

This view of the register is only valid from Armv8.1 when HCR_EL2.E2H is 1.

Any of the bits in TCR_EL2 are permitted to be cached in a TLB.

Figure A-20: AArch64_imp_atcr_el2 bit assignments

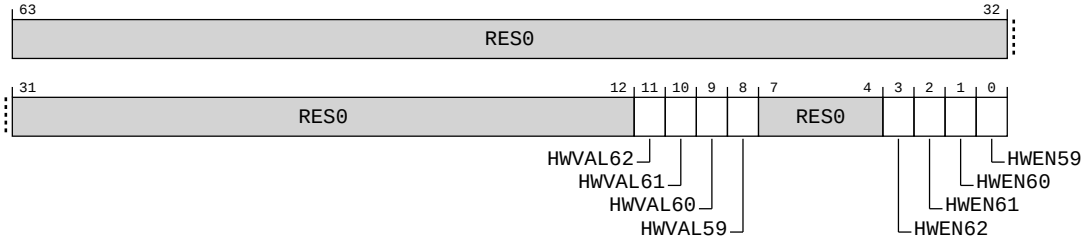


Table A-58: IMP_ATCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:12]	**RES0**	Reserved	**RES0**
[11]	HWVAL62	Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN62 is set.	0b0
[10]	HWVAL61	Value of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN61 is set.	0b0
[9]	HWVAL60	Value of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN60 is set.	0b0
[8]	HWVAL59	Value of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN59 is set.	0b0
[7:4]	**RES0**	Reserved	**RES0**
[3]	HWEN62	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[2]	HWEN61	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0

Bits	Name	Description	Reset
[1]	HWEN60	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[0]	HWEN59	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0

Access

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic IMP_ATCR_EL2 or IMP_ATCR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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

MRS <Xt>, S3_0_C15_C7_0

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

MSR S3_0_C15_C7_0, <Xt>

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

Accessibility

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic IMP_ATCR_EL2 or IMP_ATCR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.
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];
```

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);
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_ATCR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return IMP_ATCR_EL2;
    else
        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);
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_ATCR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        IMP_ATCR_EL2 = X[t];
    else
        IMP_ATCR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL1 = X[t];
```

A.1.20 IMP_AVTCR_EL2, CPU Auxiliary Virtualization 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 0000 0000 0000 0000

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-21: AArch64_imp_avtcr_el2 bit assignments

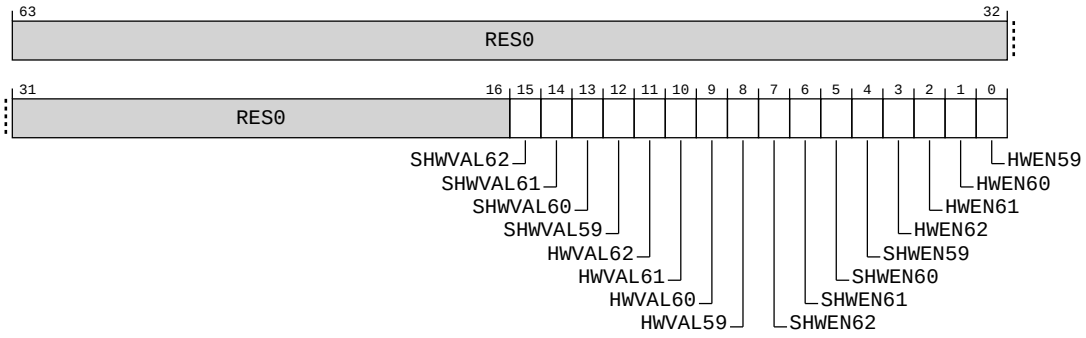


Table A-63: IMP_AVTCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15]	SHWVAL62	Value of PBHA[3] on memory accesses due to translation table walks using VSTTBR_EL2 if SHWEN62 is set.	0b0

Bits	Name	Description	Reset
[14]	SHWVAL61	Value of PBHA[2] on memory accesses due to translation table walks using VSTTBR_EL2 if SHWEN61 is set.	0b0
[13]	SHWVAL60	Value of PBHA[1] on memory accesses due to translation table walks using VSTTBR_EL2 if SHWEN60 is set.	0b0
[12]	SHWVAL59	Value of PBHA[0] on memory accesses due to translation table walks using VSTTBR_EL2 if SHWEN59 is set.	0b0
[11]	HWVAL62	Value of PBHA[3] on memory accesses due to translation table walks using VTTBR_EL2 if HWEN62 is set.	0b0
[10]	HWVAL61	Value of PBHA[2] on memory accesses due to translation table walks using VTTBR_EL2 if HWEN61 is set.	0b0
[9]	HWVAL60	Value of PBHA[1] on memory accesses due to translation table walks using VTTBR_EL2 if HWEN60 is set.	0b0
[8]	HWVAL59	Value of PBHA[0] on memory accesses due to translation table walks using VTTBR_EL2 if HWEN59 is set.	0b0
[7]	SHWEN62	Enable use of PBHA[3] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[6]	SHWEN61	Enable use of PBHA[2] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[5]	SHWEN60	Enable use of PBHA[1] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[4]	SHWEN59	Enable use of PBHA[0] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0
[3]	HWEN62	Enable use of PBHA[3] on memory accesses due to translation table walks using VTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[2]	HWEN61	Enable use of PBHA[2] on memory accesses due to translation table walks using VTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[1]	HWEN60	Enable use of PBHA[1] on memory accesses due to translation table walks using VTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[0]	HWEN59	Enable use of PBHA[0] on memory accesses due to translation table walks using VTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0

Access

MRS <Xt>, S3_4_C15_C7_1

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0111	0b001

MSR S3_4_C15_C7_1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0111	0b001

Accessibility

MRS <Xt>, S3_4_C15_C7_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if 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.1.21 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 0000 0000 0000 0000


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-22: AArch64_imp_atcr_el1 bit assignments

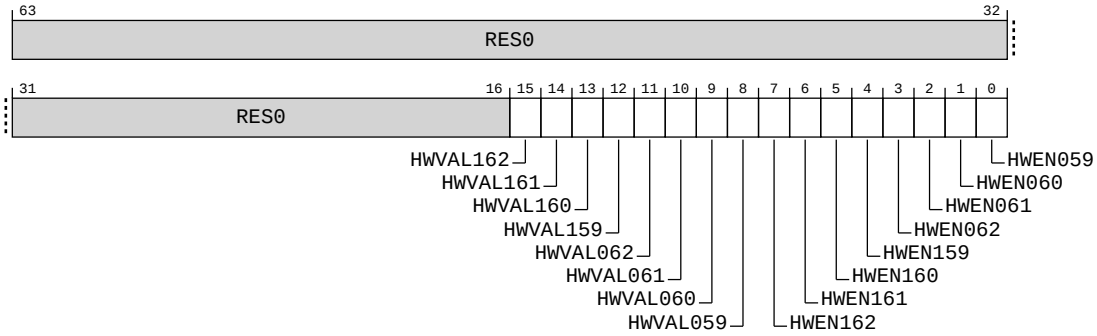


Table A-66: IMP_ATCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN162 is set.	0b0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN161 is set.	0b0
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN160 is set.	0b0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN159 is set.	0b0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL1 if HWEN062 is set.	0b0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL1 if HWEN061 is set.	0b0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL1 if HWEN060 is set.	0b0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL1 if HWEN059 is set.	0b0
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL1. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL1. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL1. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL1. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0

Access

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic IMP_ATCR_EL1 or IMP_ATCR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic IMP_ATCR_EL1 or IMP_ATCR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.
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);
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_ATCR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return IMP_ATCR_EL2;
    else
        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);
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_ATCR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        IMP_ATCR_EL2 = X[t];
    else
        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.1.22 IMP_ISIDE_DATA0_EL3, RAMINDEX Instruction Data register 0

Returns the data from a RAMINDEX instruction using RAMID=0x0,0x1

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When L1 instruction cache tag

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

When L1 instruction cache data

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When L1 instruction cache tag

Figure A-23: AArch64_imp_iside_data0_el3 bit assignments

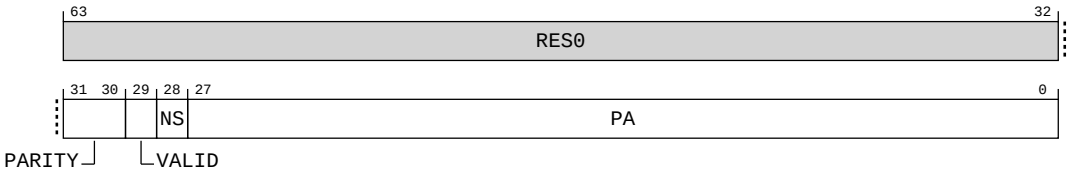


Table A-71: IMP_ISIDE_DATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:30]	PARITY	Parity bit	xx
[29]	VALID	Validity bit	x
[28]	NS	Non-secure identifier for the physical address	x

Bits	Name	Description	Reset
[27:0]	PA	Physical address [39:12]	28{x}

When L1 instruction cache data

Figure A-24: AArch64_imp_iside_data0_el3 bit assignments

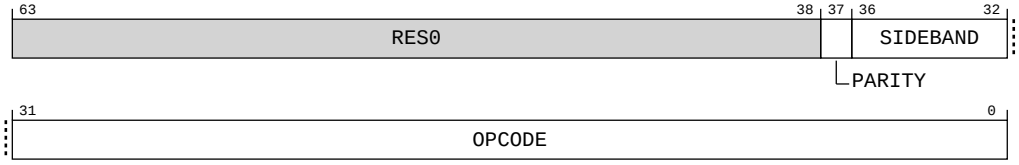


Table A-72: IMP_ISIDE_DATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:38]	RES0	Reserved	RES0
[37]	PARITY	Parity bit	x
[36:32]	SIDEBAND	Implementation defined field. If this field is not 5'b10000 or 5'b10110 then AArch64-IMP_ISIDE_DATA0_EL3.OPCODE represents the AArch64 instruction opcode	5{x}
[31:0]	OPCODE	Aarch64 opcode at {Aarch64-RAMINDEX.VA[48:3], 1'b0} if AArch64-IMP_ISIDE_DATA0_EL3.SIDEBAND is not 5'b10000 or 5'b10110	32{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_ISIDE_DATA0_EL3;
```

A.1.23 IMP_ISIDE_DATA1_EL3, RAMINDEX Instruction Data register 1

Returns the data from a RAMINDEX instruction using RAMID=0x0,0x1

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When L1 instruction cache tag

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

When L1 instruction cache data

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When L1 instruction cache tag

Figure A-25: AArch64_imp_iside_data1_el3 bit assignments

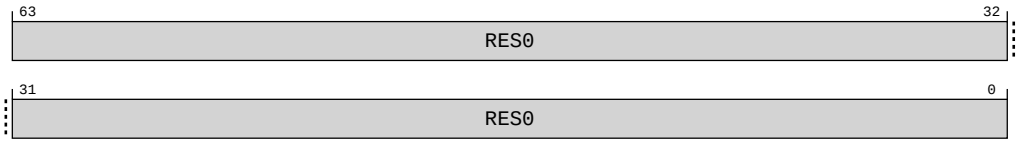


Table A-74: IMP_ISIDE_DATA1_EL3 bit descriptions

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

When L1 instruction cache data

Figure A-26: AArch64_imp_iside_data1_el3 bit assignments

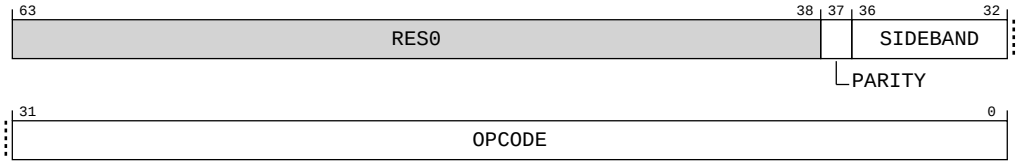


Table A-75: IMP_ISIDE_DATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:38]	RES0	Reserved	RES0
[37]	PARITY	Parity bit	x
[36:32]	SIDEBAND	Implementation defined field. If this field is not 5'b10000 or 5'b10110 then AArch64-IMP_ISIDE_DATA1_EL3.OPCODE represents the AArch64 instruction opcode	5{x}
[31:0]	OPCODE	Aarch64 opcode at {Aarch64-RAMINDEX.VA[48:3], 1'b1} if AArch64-IMP_ISIDE_DATA1_EL3.SIDEBAND is not 5'b10000 or 5'b10110	32{x}

Access

MRS <Xt>, S3_6_C15_C0_1

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0000	0b001

Accessibility

MRS <Xt>, S3_6_C15_C0_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_ISIDE_DATA1_EL3;
```

A.1.24 IMP_ISIDE_DATA2_EL3, RAMINDEX Instruction Data register 2

Returns the data from a RAMINDEX instruction using RAMID=0x0,0x1

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-27: AArch64_imp_iside_data2_el3 bit assignments

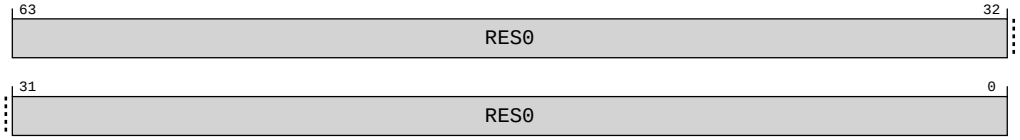


Table A-77: IMP_ISIDE_DATA2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C0_2

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0000	0b010

Accessibility

MRS <Xt>, S3_6_C15_C0_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_ISIDE_DATA2_EL3;
```

A.1.25 IMP_MMU_DATA0_EL3, RAMINDEX TLB Data register 0

Returns the data from a RAMINDEX instruction using RAMID=0x18

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When L2 TLB TCSP (small pages)

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When L2 TLB TCMP (medium pages)

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 L2 TLB TCSP (small pages)

Figure A-28: AArch64_imp_mmu_data0_el3 bit assignments

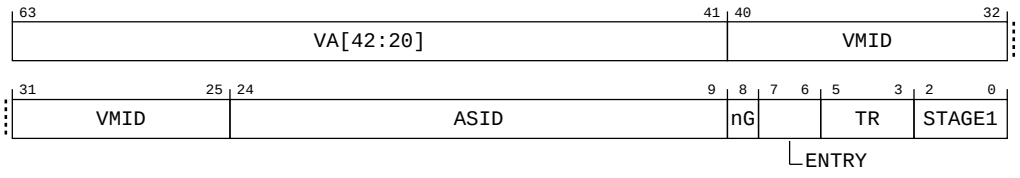


Table A-79: IMP_MMU_DATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:41]	VA[42:20]	Virtual Address [42:20]	23{x}
[40:25]	VMID	VMID	16{x}
[24:9]	ASID	ASID	16{x}
[8]	nG	Non-Global	x
[7:6]	ENTRY	Entry Size **0b00** 4KB **0b01** 16KB **0b10** 64KB **0b11** Reserved	xx
[5:3]	TR	Translation Regime **0b000** Reserved **0b001** Secure EL1 **0b010** Secure EL2 **0b011** Secure EL3 **0b100** Reserved **0b101** Non-Secure EL1 **0b110** Non-Secure EL2 **0b111** Non-Secure EL3	xxx

Bits	Name	Description	Reset
[2:0]	STAGE1	Stage 1 description encoding **0b000** Entry is found at level 3 of 4KB granule (stage1 size = 4KB) **0b001** Entry is found at level 3 of 16KB granule (stage1 size = 16KB) **0b010** Entry is found at level 3 of 64KB granule (stage1 size = 64KB) **0b011** Reserved **0b100** Entry is found at level 2 of 4KB granule (stage1 size = 2MB) **0b101** Entry is found at level 2 of 16KB granule (stage1 size = 32MB) **0b110** Entry is found at level 2 of 64KB granule (stage1 size = 512MB) **0b111** Entry is found at level 1 of 4KB granule (stage1 size = 1GB)	xxx

When L2 TLB TCMP (medium pages)

Figure A-29: AArch64_imp_mmu_data0_el3 bit assignments

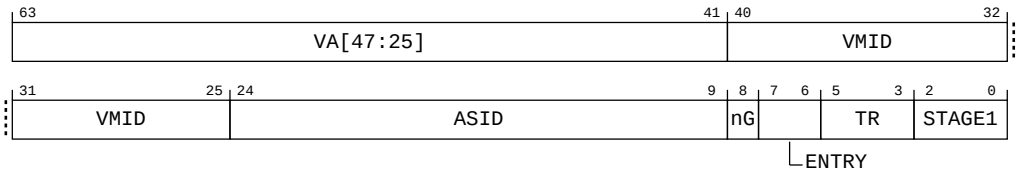


Table A-80: IMP_MMU_DATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:41]	VA[47:25]	Virtual Address [47:25]	23{x}
[40:25]	VMID	VMID	16{x}
[24:9]	ASID	ASID	16{x}
[8]	nG	Non-Global	x

Bits	Name	Description	Reset
[7:6]	ENTRY	Entry Size **0b00** 2MB **0b01** 32MB **0b10** 512MB **0b11** Normal entries: Reseved, IPA entries: 2MB originating from a 1GB block	xx
[5:3]	TR	Translation Regime **0b000** Reserved **0b001** Secure EL1 **0b010** Secure EL2 **0b011** Secure EL3 **0b100** Reserved **0b101** Non-Secure EL1 **0b110** Non-Secure EL2 **0b111** Non-Secure EL3	xxx

Bits	Name	Description	Reset
[2:0]	STAGE1	Stage 1 description encoding **0b000** Entry is found at level 3 of 4KB granule (stage1 size = 4KB) **0b001** Entry is found at level 3 of 16KB granule (stage1 size = 16KB) **0b010** Entry is found at level 3 of 64KB granule (stage1 size = 64KB) **0b011** Entry is an IPA to PA translation **0b100** Entry is found at level 2 of 4KB granule (stage1 size = 2MB) **0b101** Entry is found at level 2 of 16KB granule (stage1 size = 32MB) **0b110** Entry is found at level 2 of 64KB granule (stage1 size = 512MB) **0b111** Entry is found at level 1 of 4KB granule (stage1 size = 1GB)	xxx

Access

MRS <Xt>, S3_6_C15_C0_3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0000	0b011

Accessibility

MRS <Xt>, S3_6_C15_C0_3

```
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_MMU_DATA0_EL3;
```

A.1.26 IMP_MMU_DATA1_EL3, RAMINDEX TLB Data register 1

Returns the data from a RAMINDEX instruction using RAMID=0x18

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When L2 TLB TCSP (small pages)

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When L2 TLB TCMP (medium pages) and AArch64-IMP_MMU_DATA1_EL3.CACHEABLITY == 0b0

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When L2 TLB TCMP (medium pages) and AArch64-IMP_MMU_DATA1_EL3.CACHEABLITY == 0b1

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 L2 TLB TCSP (small pages)

Figure A-30: AArch64_imp_mmu_data1_el3 bit assignments

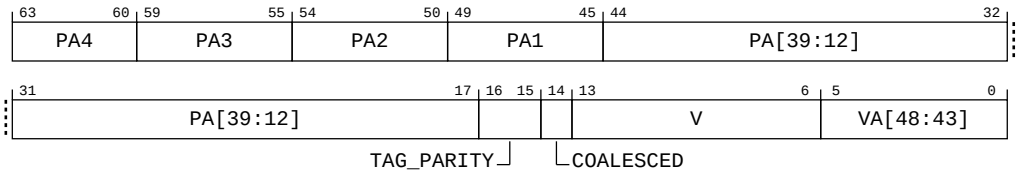


Table A-82: IMP_MMU_DATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:60]	PA4	Physical Address cluster 4[3:0]	xxxx
[59:55]	PA3	Physical Address cluster 3	5{x}
[54:50]	PA2	Physical Address cluster 2	5{x}
[49:45]	PA1	Physical address cluster 1	5{x}
[44:17]	PA[39:12]	Physical address [39:12]	28{x}
[16:15]	TAG_PARITY	Tag Parity	xx

Bits	Name	Description	Reset
[14]	COALESCED	Coalesced entry	x
[13:6]	V	Validity bits	8{x}
[5:0]	VA[48:43]	Virtual Address [48:43]	6{x}

When L2 TLB TCMP (medium pages) and AArch64-IMP_MMU_DATA1_EL3.CACHEABLITY == 0b0

Figure A-31: AArch64_imp_mmu_data1_el3 bit assignments

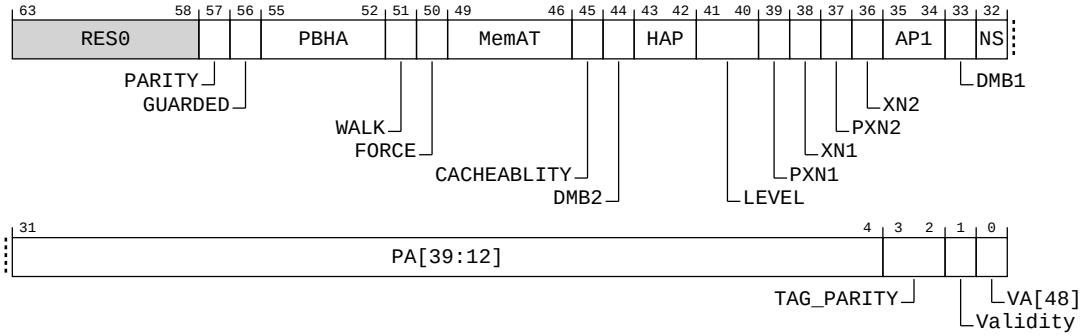


Table A-83: IMP_MMU_DATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:58]	RES0	Reserved	RES0
[57]	PARITY	Data Parity	x
[56]	GUARDED	Guarded page	x
[55:52]	PBHA	PBHA	xxxx
[51]	WALK	Walk cache entry	x
[50]	FORCE	Force write-back	x

Bits	Name	Description	Reset
[49:46]	MemAT	Memory attributes information **0b0000** Normal memory - Non-Cacheable due to stage 1. Originnaly not outer cacheable (WB/ WT) **0b0001** Device nGnRnE with stage 1 being device **0b0010** Device nGnRnE with stage 1 being normal non-cacheable **0b0011** Device nGnRnE with stage 1 being normal cacheable **0b0100** Normal memory - Non-Cacheable due to stage 1. Originnaly outer cacheable (WB/ WT) **0b0101** Device nGnRE with stage 1 being device **0b0110** Device nGnRE with stage 1 being normal non-cacheable **0b0111** Device nGnRE with stage 1 being normal cacheable **0b1000** Normal memory - Non-Cacheable due to stage 2. Originnaly not outer cacheable (WB/ WT) **0b1001** Device nGRE with stage 1 being device **0b1010** Device nGRE with stage 1 being normal non-cacheable **0b1011** Device nGRE with stage 1 being normal cacheable **0b1100** Normal memory - Non-Cacheable due to stage 2. Originnaly outer cacheable (WB/ WT) **0b1101** Device GRE with stage 1 being device **0b1110** Device GRE with stage 1 being normal non-cacheable **0b1111** Device GRE with stage 1 being normal cacheable	xxxx
[45]	CACHEABLITY	Cacheability	x
[44]	DMB2	Stage 2 dirty bit Modifier	x
[43:42]	HAP	Stage 2 Access Permissions	xx

Bits	Name	Description	Reset
[41:40]	LEVEL	Stage 2 level **0b00** Level 3 **0b01** Level 2 **0b10** Level 1 **0b11** Level 0	xx
[39]	PXN1	Stage 1 privileged Execute-Never	x
[38]	XN1	Stage 1 Execute-Never	x
[37]	PXN2	Stage 2 privileged Execute-Never	x
[36]	XN2	Stage 2 Execute-Never	x
[35:34]	AP1	Stage 1 Access Permission	xx
[33]	DMB1	Stage 1 dirty bit Modifier	x
[32]	NS	Non-Secure which determines whether the physical address is in secure spece or non-secure space	x
[31:4]	PA[39:12]	Physical Address[39:12]	28{x}
[3:2]	TAG_PARITY	Tag Parity	xx
[1]	Validity	Valididy	x
[0]	VA[48]	Virtual Address [48]	x

When L2 TLB TCMP (medium pages) and AArch64-IMP_MMU_DATA1_EL3.CACHEABLITY == 0b1

Figure A-32: AArch64_imp_mmu_data1_el3 bit assignments

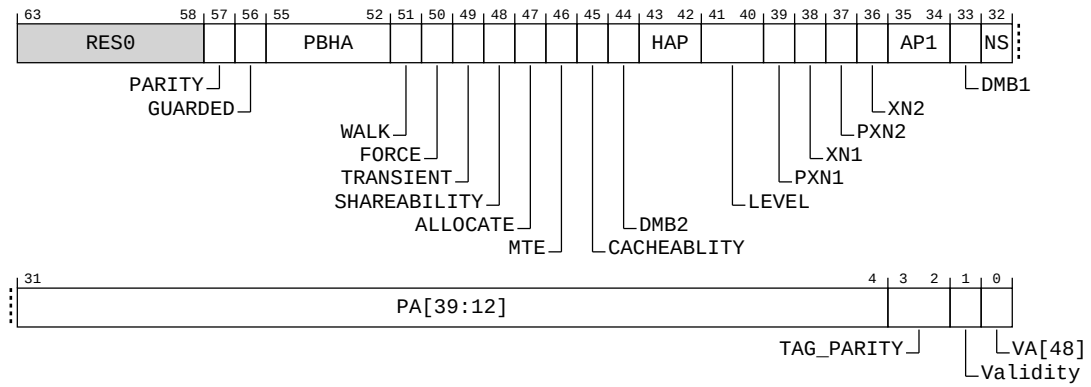


Table A-84: IMP_MMU_DATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:58]	**RES0**	Reserved	**RES0**
[57]	PARITY	Data Parity	x
[56]	GUARDED	Guarded page	x

Bits	Name	Description	Reset
[55:52]	PBHA	PBHA	xxxx
[51]	WALK	Walk cache entry	x
[50]	FORCE	Force write-back	x
[49]	TRANSIENT	Inner transient hint	x
[48]	SHAREABILITY	Shareability	x
[47]	ALLOCATE	Allocate hint	x
[46]	MTE	MTE tag	x
[45]	CACHEABLITY	Cacheability	x
[44]	DMB2	Stage 2 dirty bit Modifier	x
[43:42]	HAP	Stage 2 Access Permissions	xx
[41:40]	LEVEL	Stage 2 level **0b00** Level 3 **0b01** Level 2 **0b10** Level 1 **0b11** Level 0	xx
[39]	PXN1	Stage 1 privileged Execute-Never	x
[38]	XN1	Stage 1 Execute-Never	x
[37]	PXN2	Stage 2 privileged Execute-Never	x
[36]	XN2	Stage 2 Execute-Never	x
[35:34]	AP1	Stage 1 Access Permission	xx
[33]	DMB1	Stage 1 dirty bit Modifier	x
[32]	NS	Non-Secure which determines whether the physical address is in secure spece or non-secure space	x
[31:4]	PA[39:12]	Physical Address[39:12]	28{x}
[3:2]	TAG_PARITY	Tag Parity	xx
[1]	Validity	Valididy	x
[0]	VA[48]	Virtual Address [48]	x

Access

MRS <Xt>, S3_6_C15_C0_4

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0000	0b100

Accessibility

MRS <Xt>, S3_6_C15_C0_4

```
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_MMU_DATA1_EL3;
```

A.1.27 IMP_MMU_DATA2_EL3, RAMINDEX TLB Data register 2

Returns the data from a RAMINDEX instruction using RAMID=0x18

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When L2 TLB TCSP (small pages) and AArch64-IMP_MMU_DATA1_EL3.CACHEABLITY == 0b0

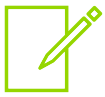
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When L2 TLB TCSP (small pages) and AArch64-IMP_MMU_DATA1_EL3.CACHEABLITY == 0b1

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When L2 TLB TCMP (medium pages)

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 L2 TLB TCSP (small pages) and AArch64-IMP_MMU_DATA1_EL3.CACHEABLITY == 0b0

Figure A-33: AArch64_imp_mmu_data2_el3 bit assignments

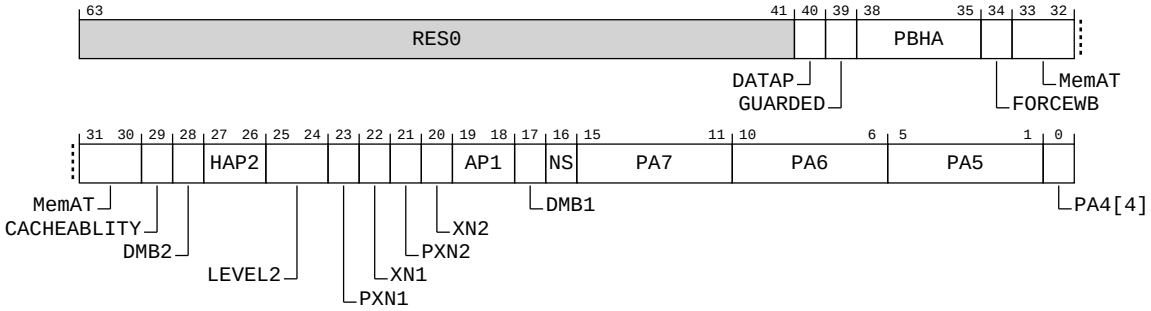


Table A-86: IMP_MMU_DATA2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:41]	RES0	Reserved	RES0
[40]	DATAP	Data Parity	x
[39]	GUARDED	Guarded page	x
[38:35]	PBHA	PBHA	xxxx
[34]	FORCEWB	Force write-back	x

Bits	Name	Description	Reset
[33:30]	MemAT	Memory attributes information **0b0000** Normal memory - Non-Cacheable due to stage 1. Originnaly not outer cacheable (WB/ WT) **0b0001** Device nGnRnE with stage 1 being device **0b0010** Device nGnRnE with stage 1 being normal non-cacheable **0b0011** Device nGnRnE with stage 1 being normal cacheable **0b0100** Normal memory - Non-Cacheable due to stage 1. Originnaly outer cacheable (WB/ WT) **0b0101** Device nGnRE with stage 1 being device **0b0110** Device nGnRE with stage 1 being normal non-cacheable **0b0111** Device nGnRE with stage 1 being normal cacheable **0b1000** Normal memory - Non-Cacheable due to stage 2. Originnaly not outer cacheable (WB/ WT) **0b1001** Device nGRE with stage 1 being device **0b1010** Device nGRE with stage 1 being normal non-cacheable **0b1011** Device nGRE with stage 1 being normal cacheable **0b1100** Normal memory - Non-Cacheable due to stage 2. Originnaly outer cacheable (WB/ WT) **0b1101** Device GRE with stage 1 being device **0b1110** Device GRE with stage 1 being normal non-cacheable **0b1111** Device GRE with stage 1 being normal cacheable	xxxx
[29]	CACHEABLITY	Cacheability	x
[28]	DMB2	Stage 2 dirty bit Modifier	x
[27:26]	HAP2	Stage 2 Access Permissions	xx

Bits	Name	Description	Reset
[25:24]	LEVEL2	Stage 2 level **0b00** Level 3 **0b01** Level 2 **0b10** Level 1 **0b11** Level 0	xx
[23]	PXN1	Stage 1 privileged Execute-Never	x
[22]	XN1	Stage 1 Execute-Never	x
[21]	PXN2	Stage 2 privileged Execute-Never	x
[20]	XN2	Stage 2 Execute-Never	x
[19:18]	AP1	Stage 1 Access Permission	xx
[17]	DMB1	Stage 1 dirty bit Modifier	x
[16]	NS	Non-Secure which determines whether the physical address is in secure spece or non-secure space	x
[15:11]	PA7	Physical Address cluster 7	5{x}
[10:6]	PA6	Physical Address cluster 6	5{x}
[5:1]	PA5	Physical Address cluster 5	5{x}
[0]	PA4[4]	Physical Address cluster 4 bit 4	x

When L2 TLB TCSP (small pages) and AArch64-IMP_MMU_DATA1_EL3.CACHEABLITY == 0b1

Figure A-34: AArch64_imp_mmu_data2_el3 bit assignments

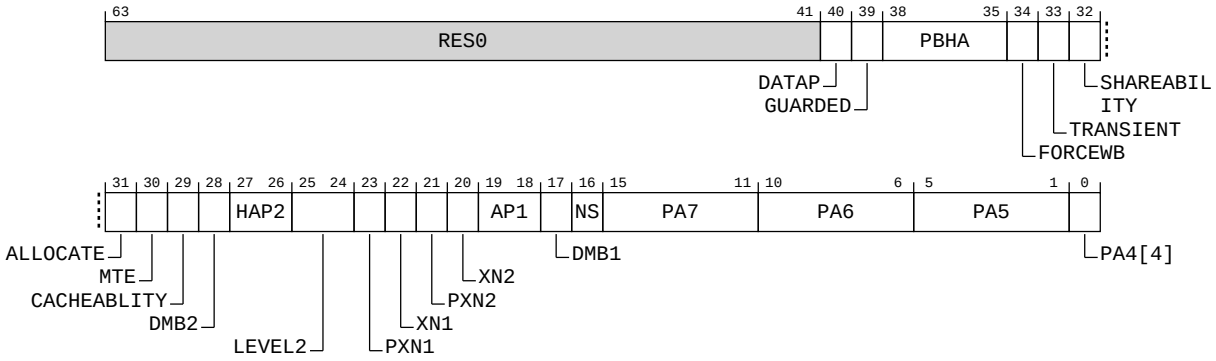


Table A-87: IMP_MMU_DATA2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:41]	**RES0**	Reserved	**RES0**
[40]	DATAP	Data Parity	x
[39]	GUARDED	Guarded page	x
[38:35]	PBHA	PBHA	xxxx

Bits	Name	Description	Reset
[34]	FORCEWB	Force write-back	x
[33]	TRANSIENT	Inner transient hint	x
[32]	SHAREABILITY	Shareability	x
[31]	ALLOCATE	Allocate hint	x
[30]	MTE	MTE tag	x
[29]	CACHEABLITY	Cacheability	x
[28]	DMB2	Stage 2 dirty bit Modifier	x
[27:26]	HAP2	Stage 2 Access Permissions	xx
[25:24]	LEVEL2	Stage 2 level **0b00** Level 3 **0b01** Level 2 **0b10** Level 1 **0b11** Level 0	xx
[23]	PXN1	Stage 1 privileged Execute-Never	x
[22]	XN1	Stage 1 Execute-Never	x
[21]	PXN2	Stage 2 privileged Execute-Never	x
[20]	XN2	Stage 2 Execute-Never	x
[19:18]	AP1	Stage 1 Access Permission	xx
[17]	DMB1	Stage 1 dirty bit Modifier	x
[16]	NS	Non-Secure which determines whether the physical address is in secure spece or non-secure space	x
[15:11]	PA7	Physical Address cluster 7	5{x}
[10:6]	PA6	Physical Address cluster 6	5{x}
[5:1]	PA5	Physical Address cluster 5	5{x}
[0]	PA4[4]	Physical Address cluster 4 bit 4	x

When L2 TLB TCMP (medium pages)

Figure A-35: AArch64_imp_mmu_data2_el3 bit assignments

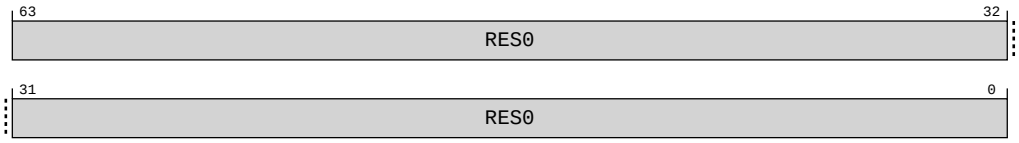


Table A-88: IMP_MMU_DATA2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C0_5

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0000	0b101

Accessibility

MRS <Xt>, S3_6_C15_C0_5

```
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_MMU_DATA2_EL3;
```

A.1.28 IMP_DSIDE_DATA0_EL3, RAMINDEX L1D Data register 0

Returns the data from a RAMINDEX instruction using RAMID=0x8,0x9

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When L1 Data cache tag

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When L1 data cache data

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When L1 Data cache tag

Figure A-36: AArch64_imp_dside_data0_el3 bit assignments

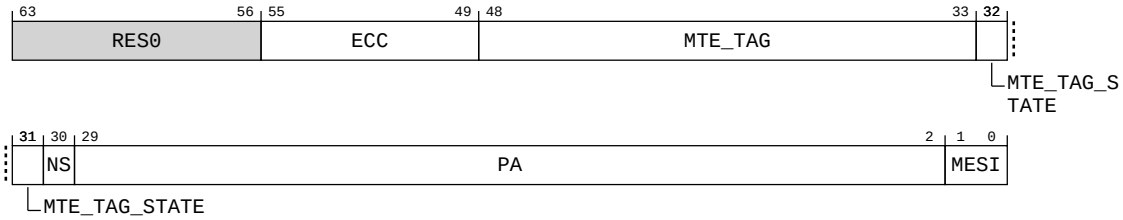


Table A-90: IMP_DSIDE_DATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	RES0
[55:49]	ECC	ECC	7{x}
[48:33]	MTE_TAG	MTE tag data	16{x}
[32:31]	MTE_TAG_STATE	MTE tag state **0b00** Invalid **0b01** Invalid **0b10** Clean **0b11** Dirty state	xx
[30]	NS	Non-secure identifier for the physical address	x
[29:2]	PA	Physical address [39:12]	28{x}
[1:0]	MESI	MESI **0b00** Invalid **0b01** Shared **0b10** Modified (unique dirty) **0b11** Exclusive (unique clean)	xx

When L1 data cache data

Figure A-37: AArch64_imp_dside_data0_el3 bit assignments

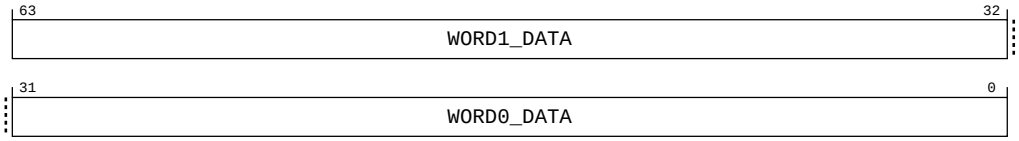


Table A-91: IMP_DSIDE_DATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:32]	WORD1_DATA	Word 1 data 31:0	32{x}
[31:0]	WORD0_DATA	Word 0 data 31:0	32{x}

Access

MRS <Xt>, S3_6_C15_C1_0

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0001	0b000

Accessibility

MRS <Xt>, S3_6_C15_C1_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_DSIDE_DATA0_EL3;
```

A.1.29 IMP_DSIDE_DATA1_EL3, RAMINDEX L1D Data register 1

Returns the data from a RAMINDEX instruction using RAMID=0x8,0x9

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When L1 data cache tag

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When L1 data cache data

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When L1 data cache tag

Figure A-38: AArch64_imp_dside_data1_el3 bit assignments

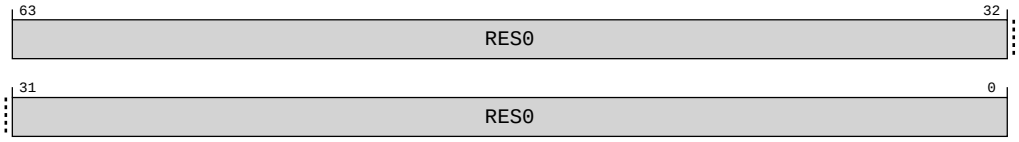


Table A-93: IMP_DSIDE_DATA1_EL3 bit descriptions

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

When L1 data cache data

Figure A-39: AArch64_imp_dside_data1_el3 bit assignments

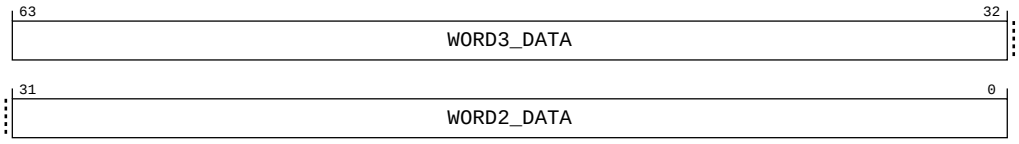


Table A-94: IMP_DSIDE_DATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:32]	WORD3_DATA	Word 3 data 31:0	32{x}
[31:0]	WORD2_DATA	Word 2 data 31:0	32{x}

Access

MRS <Xt>, S3_6_C15_C1_1

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0001	0b001

Accessibility

MRS <Xt>, S3_6_C15_C1_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_DSIDE_DATA1_EL3;
```

A.1.30 IMP_DSIDE_DATA2_EL3, RAMINDEX L1D Data register 2

Returns the data from a RAMINDEX instruction using RAMID=0x8,0x9

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When L1 data cache tag

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

When L1 data cache data

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When L1 data cache tag

Figure A-40: AArch64_imp_dside_data2_el3 bit assignments

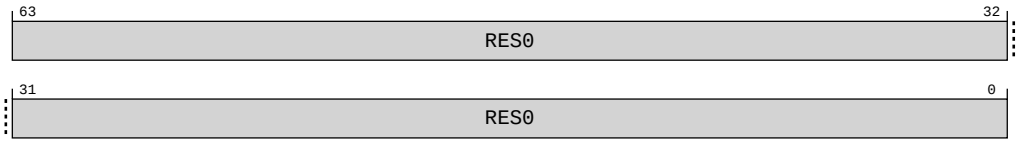


Table A-96: IMP_DSIDE_DATA2_EL3 bit descriptions

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

When L1 data cache data

Figure A-41: AArch64_imp_dside_data2_el3 bit assignments

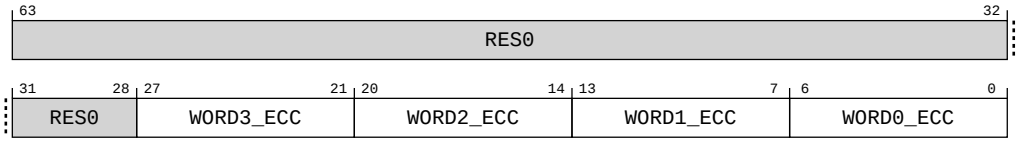


Table A-97: IMP_DSIDE_DATA2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:28]	**RES0**	Reserved	**RES0**
[27:21]	WORD3_ECC	Word 3 ECC	7{x}
[20:14]	WORD2_ECC	Word 2 ECC	7{x}
[13:7]	WORD1_ECC	Word 1 ECC	7{x}
[6:0]	WORD0_ECC	Word 0 ECC	7{x}

Access

MRS <Xt>, S3_6_C15_C1_2

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0001	0b010

Accessibility

MRS <Xt>, S3_6_C15_C1_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_DSIDE_DATA2_EL3;
```

A.1.31 IMP_L2_DATA0_EL3, RAMINDEX L2 Data register 0

Returns the data from a RAMINDEX instruction using RAMID=0x10 or RAMID=0x11

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When L2 tag cache for 512KB RAM

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When L2 tag cache for 256KB RAM

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When L2 tag cache for 128KB RAM

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

When L2 data cache

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 L2 tag cache for 512KB RAM

Figure A-42: AArch64_imp_l2_data0_el3 bit assignments

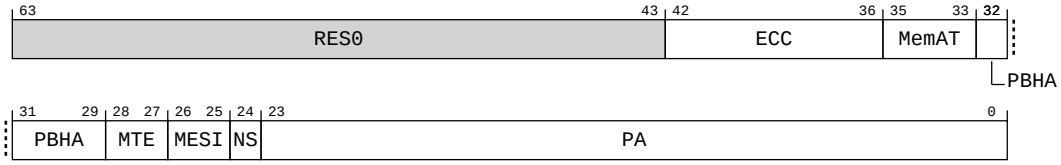


Table A-99: IMP_L2_DATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:43]	RES0	Reserved	RES0
[42:36]	ECC	ECC	7{x}
[35:33]	MemAT	Memory attribute	xxx
[32:29]	PBHA	PBHA	xxxx
[28:27]	MTE	MTE state **0b00** Invalid **0b01** Invalid **0b10** Clean **0b11** Dirty	xx
[26:25]	MESI	MESI **0b00** Invalid **0b01** Shared **0b10** Modified (unique dirty) **0b11** Exclusice (unique clean)	xx
[24]	NS	Non-Secure indentifier	x
[23:0]	PA	Physical address [39:16]	24{x}

When L2 tag cache for 256KB RAM

Figure A-43: AArch64_imp_l2_data0_el3 bit assignments

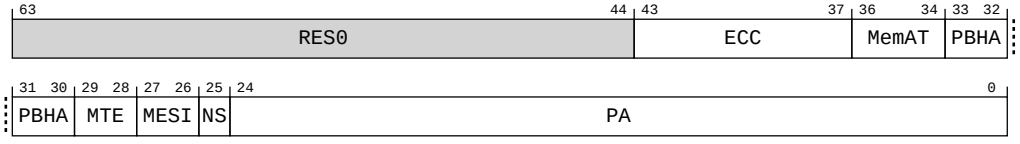


Table A-100: IMP_L2_DATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:44]	RES0	Reserved	RES0
[43:37]	ECC	ECC	7{x}
[36:34]	MemAT	Memory attribute	xxx
[33:30]	PBHA	PBHA	xxxx
[29:28]	MTE	MTE state **0b00** Invalid **0b01** Invalid **0b10** Clean **0b11** Dirty	xx
[27:26]	MESI	MESI **0b00** Invalid **0b01** Shared **0b10** Modified (unique dirty) **0b11** Exclusice (unique clean)	xx
[25]	NS	Non-Secure indentifier	x
[24:0]	PA	Physical address [39:15]	25{x}

When L2 tag cache for 128KB RAM

Figure A-44: AArch64_imp_l2_data0_el3 bit assignments

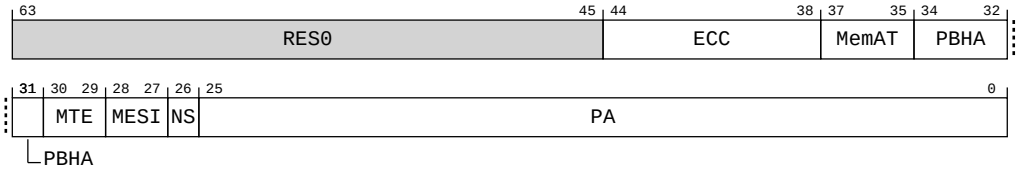


Table A-101: IMP_L2_DATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:45]	RES0	Reserved	RES0
[44:38]	ECC	ECC	7{x}
[37:35]	MemAT	Memory attribute	xxx
[34:31]	PBHA	PBHA	xxxx

Bits	Name	Description	Reset
[30:29]	MTE	MTE state **0b00** Invalid **0b01** Invalid **0b10** Clean **0b11** Dirty	xx
[28:27]	MESI	MESI **0b00** Invalid **0b01** Shared **0b10** Modified (unique dirty) **0b11** Exclusice (unique clean)	xx
[26]	NS	Non-Secure indentifier	x
[25:0]	PA	Physical address [39:14]	26{x}

When L2 data cache

Figure A-45: AArch64_imp_l2_data0_el3 bit assignments

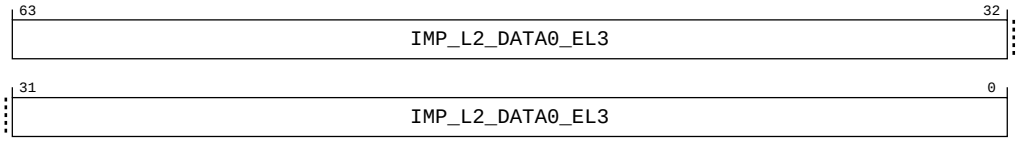


Table A-102: IMP_L2_DATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	IMP_L2_DATA0_EL3	Least significant 8-bytes of the 16-bytes data granule of the line	64{x}

Access

MRS <Xt>, S3_6_C15_C1_3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0001	0b011

Accessibility

MRS <Xt>, S3_6_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
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_L2_DATA0_EL3;
```

A.1.32 IMP_L2_DATA2_EL3, RAMINDEX L2 Data register 2

Returns the data from a RAMINDEX instruction using RAMID=0x10 or RAMID=0x11

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When L2 tag cache

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

When L2 data cache

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 L2 tag cache

Figure A-46: AArch64_imp_l2_data2_el3 bit assignments

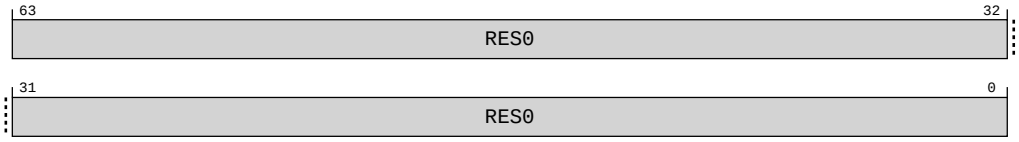


Table A-104: IMP_L2_DATA2_EL3 bit descriptions

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

When L2 data cache

Figure A-47: AArch64_imp_l2_data2_el3 bit assignments

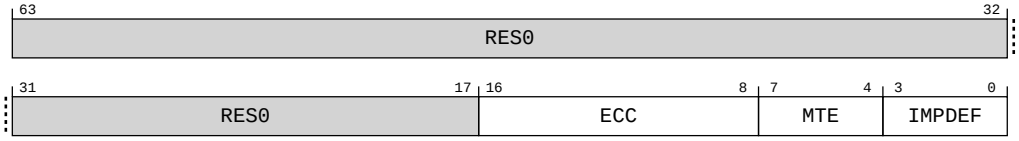


Table A-105: IMP_L2_DATA2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:17]	**RES0**	Reserved	**RES0**
[16:8]	ECC	ECC	9{x}
[7:4]	MTE	MTE allocation tags for the 16-bytes granule	xxxx
[3:0]	IMPDEF	For granule 0 and 3: Implementation defined meta data For granule 1 and 2: MPAM	xxxx

Access

MRS <Xt>, S3_6_C15_C1_4

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0001	0b100

Accessibility

MRS <Xt>, S3_6_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
        UNDEFINED;
elsif PSTATE.EL == EL2 then
```

```
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_L2_DATA2_EL3;
```

A.1.33 IMP_L2_DATA1_EL3, RAMINDEX L2 Data register 1

Returns the data from a RAMINDEX instruction using RAMID=0x10 or RAMID=0x11

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When L2 tag cache

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

When L2 data cache

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 L2 tag cache

Figure A-48: AArch64_imp_l2_data1_el3 bit assignments

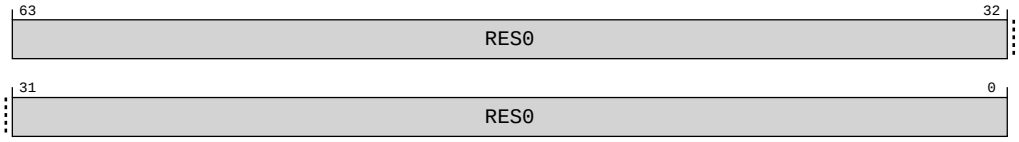


Table A-107: IMP_L2_DATA1_EL3 bit descriptions

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

When L2 data cache

Figure A-49: AArch64_imp_l2_data1_el3 bit assignments

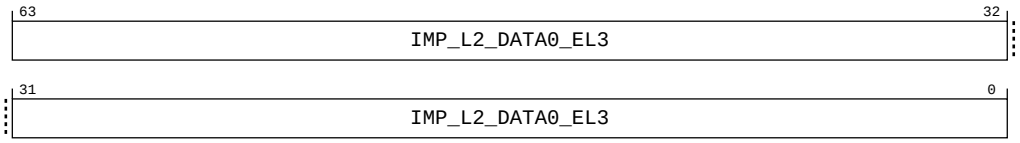


Table A-108: IMP_L2_DATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	IMP_L2_DATA0_EL3	Most significant 8-bytes of the 16-bytes data granule of the line	`64{x}`

Access

MRS <Xt>, S3_6_C15_C1_5

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0001	0b101

Accessibility

MRS <Xt>, S3_6_C15_C1_5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_L2_DATA1_EL3;
```

A.1.34 IMP_CLUSTERCDBG_EL3, Cluster Cache Debug Register

Can be used to read the contents of the L3 cache RAMs and snoop filter RAMs. The register must be written with the information of which RAM is to be read. Then the same register should be read to read the contents of that RAM.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Bit descriptions

Figure A-50: AArch64_imp_clustercdbg_el3 bit assignments

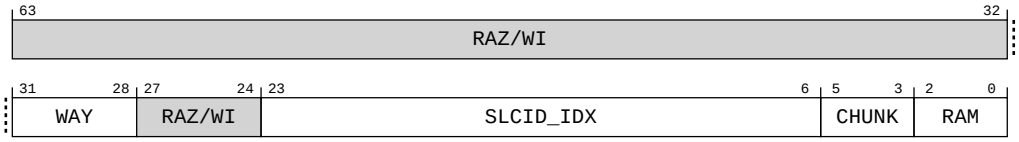


Table A-110: IMP_CLUSTERCDBG_EL3 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RAZ/WI**	Reserved	**RAZ/WI**
[31:28]	WAY	Way of RAM being accessed.	0b0000
[27:24]	**RAZ/WI**	Reserved	**RAZ/WI**
[23:6]	SLCID_IDX	The L3 cache Set locations in each cache slice are all power-of-2 in size and therefore can be identified using contiguous index locations. The Set index values for slice 0 start from value zero in this field, followed by the index locations for slice 1, then slice 2, and so on. The total index width varies depending on the size of the RAM being accessed. The cache slice identification number, Slice ID, forms the upper used bits of the cache location encoding in this field. For a Tag RAM or Data RAM access this field will encode as {'0, SLICE_ID_W, TagRAM_IDX_W} For a Snoop Filter RAM access this field will encode as {'0, SLICE_ID_W, SFRAM_IDX_W}.	0b00000000000000000000

Bits	Name	Description	Reset
[5:3]	CHUNK	Select of 64-bit data chunk to read from 512-bit Data RAM cache line. Only used when accessing Data RAM data. **0b000** Data[63:0] **0b001** Data[127:64] **0b010** Data[191:128] **0b011** Data[255:192] **0b100** Data[319:256] **0b101** Data[383:320] **0b110** Data[447:384] **0b111** Data[511:448]	0b000
[2:0]	RAM	RAM to be accessed. All other values are reserved. **0b001** Snoop Filter RAM **0b010** Tag RAM **0b011** Data RAM - accessing cacheline data **0b111** Data RAM - accessing cacheline MTE tags	0b000

Access

MRS <Xt>, S3_6_C15_C4_7

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0100	0b111

MSR S3_6_C15_C4_7, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0100	0b111

Accessibility

MRS <Xt>, S3_6_C15_C4_7

```
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_CLUSTERCDBG_EL3;
```

MSR S3_6_C15_C4_7, <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_CLUSTERCDBG_EL3 = X[t];
```

A.1.35 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 0000 xxxx 0000


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-51: AArch64_imp_atcr_el3 bit assignments

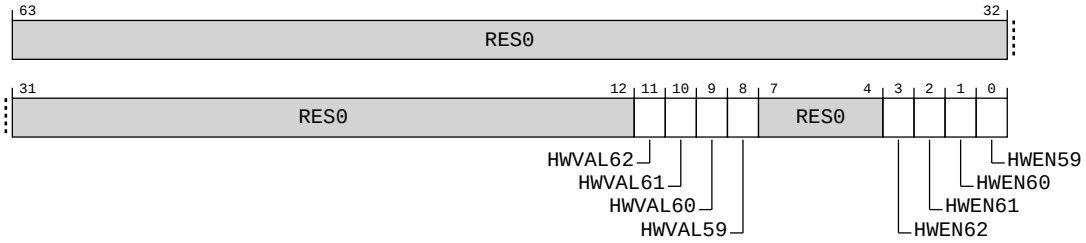


Table A-113: IMP_ATCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:12]	**RES0**	Reserved	**RES0**
[11]	HWVAL62	Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN62 is set.	0b0
[10]	HWVAL61	Value of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN61 is set.	0b0
[9]	HWVAL60	Value of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN60 is set.	0b0
[8]	HWVAL59	Value of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN59 is set.	0b0
[7:4]	**RES0**	Reserved	**RES0**
[3]	HWEN62	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[2]	HWEN61	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[1]	HWEN60	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[0]	HWEN59	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0

Access

MRS <Xt>, S3_6_C15_C7_0

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0111	0b000

MSR S3_6_C15_C7_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0111	0b000

Accessibility

MRS <Xt>, S3_6_C15_C7_0

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

```
    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.1.36 IMP_CPUPSELR_EL3, Selected Instruction Private Control Register

Selects the current instruction patch register for subsequent accesses to AArch64-IMP_CPUPCR_EL3, AArch64-IMP_CPUPOR_EL3, AArch64-IMP_CPUPMR_EL3, AArch64-IMP_CPUPOR2_EL3, AArch64-IMP_CPUPMR2_EL3, and AArch64-IMP_CPUPFR_EL3

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-52: AArch64_imp_cpupselr_el3 bit assignments

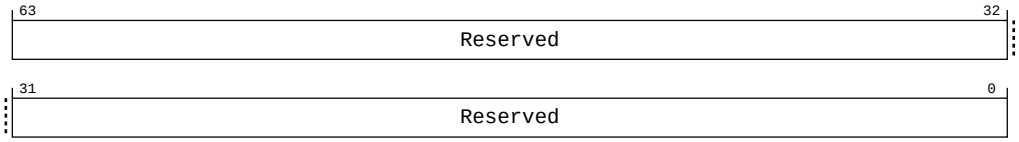


Table A-116: IMP_CPUPSELR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_0

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b000

MSR S3_6_C15_C8_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b000

Accessibility

MRS <Xt>, S3_6_C15_C8_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_CPUPSELR_EL3;
```

MSR S3_6_C15_C8_0, <Xt>

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

```
IMP_CPUPSELR_EL3 = X[t];
```

A.1.37 IMP_CPUPCR_EL3, Selected Instruction Private Control Register

Configures current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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_imp_cpupcr_el3 bit assignments

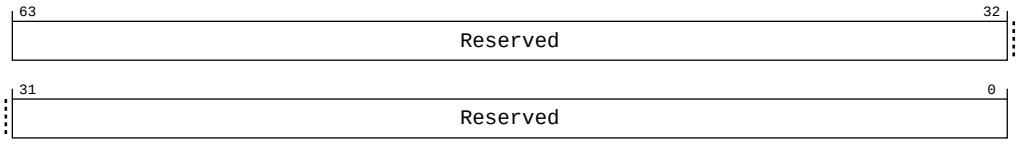


Table A-119: IMP_CPUPCR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_1

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b001

MSR S3_6_C15_C8_1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b001

Accessibility

MRS <Xt>, S3_6_C15_C8_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_CPUPCR_EL3;
```

MSR S3_6_C15_C8_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPCR_EL3 = X[t];
```

A.1.38 IMP_CPUPOR_EL3, Selected Instruction Patch Opcode Register

Opcode for current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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_imp_cpupor_el3 bit assignments

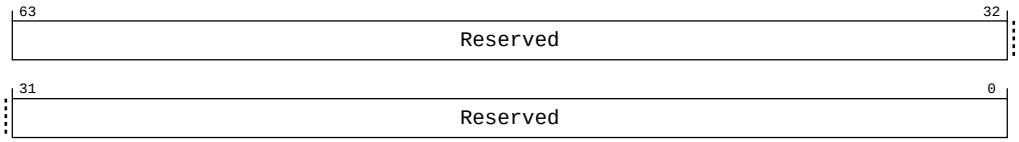


Table A-122: IMP_CPUPOR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_2

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b010

MSR S3_6_C15_C8_2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b010

Accessibility

MRS <Xt>, S3_6_C15_C8_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_CPUPOR_EL3;
```

MSR S3_6_C15_C8_2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPOR_EL3 = X[t];
```

A.1.39 IMP_CPUPMR_EL3, Selected Instruction Patch Mask Register

Mask for current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-55: AArch64_imp_cpupmr_el3 bit assignments

Table A-125: IMP_CPUPMR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b011

MSR S3_6_C15_C8_3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b011

Accessibility

MRS <Xt>, S3_6_C15_C8_3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_CPUPMR_EL3;
```

MSR S3_6_C15_C8_3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPMR_EL3 = X[t];
```

A.1.40 IMP_CPUPOR2_EL3, Selected Instruction Patch Opcode Register 2

Opcode exclusion for current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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_imp_cpupor2_el3 bit assignments

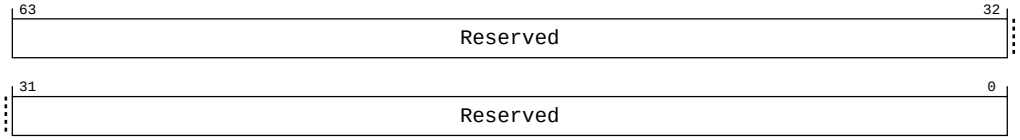


Table A-128: IMP_CPUPOR2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_4

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b100

MSR S3_6_C15_C8_4, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b100

Accessibility

MRS <Xt>, S3_6_C15_C8_4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_CPUPOR2_EL3;
```

MSR S3_6_C15_C8_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
    IMP_CPUPOR2_EL3 = X[t];
```

A.1.41 IMP_CPUPMR2_EL3, Selected Instruction Private Mask Register 2

Mask exclusion for current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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_imp_cpupmr2_el3 bit assignments

Table A-131: IMP_CPUPMR2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_5

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b101

MSR S3_6_C15_C8_5, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b101

Accessibility

MRS <Xt>, S3_6_C15_C8_5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_CPUPMR2_EL3;
```

MSR S3_6_C15_C8_5, <Xt>

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

```
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPMR2_EL3 = X[t];
```

A.1.42 IMP_CPUPFR_EL3, Selected Instruction Private Flag Register

Instruction Patch flags for current Instruction Patch selected by AArch64-IMP_CPUPSELR_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

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-58: AArch64_imp_cpupfr_el3 bit assignments

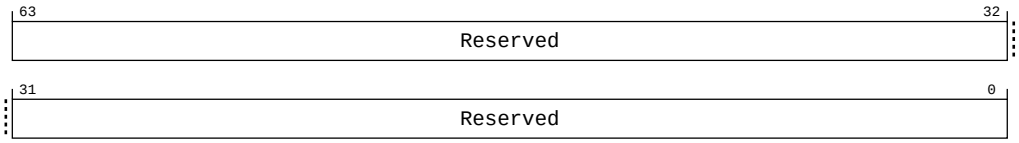


Table A-134: IMP_CPUPFR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_6

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b110

MSR S3_6_C15_C8_6, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b1000	0b110

Accessibility

MRS <Xt>, S3_6_C15_C8_6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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_CPUPFR_EL3;
```

MSR S3_6_C15_C8_6, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPFR_EL3 = X[t];
```

A.1.43 FPCR, Floating-point Control Register

Controls floating-point behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x000 0000 0000 xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-59: AArch64_fpcr bit assignments

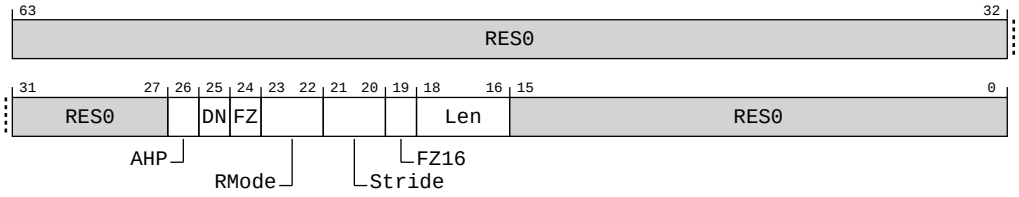


Table A-137: FPCR bit descriptions

Bits	Name	Description	Reset
[63:27]	RES0	Reserved	RES0
[26]	AHP	Alternative half-precision control bit. **0b0** IEEE half-precision format selected. **0b1** Alternative half-precision format selected. This bit is used only for conversions between half-precision floating-point and other floating-point formats. The data-processing instructions added as part of the FEAT_FP16 extension always use the IEEE half-precision format, and ignore the value of this bit.	0b0

Bits	Name	Description	Reset
[25]	DN	Default NaN use for NaN propagation. **0b0** NaN operands propagate through to the output of a floating-point operation. **0b1** Any operation involving one or more NaNs returns the Default NaN. This bit has no effect on the output of FABS, FMAX*, FMIN*, and FNEG instructions, and a default NaN is never returned as a result of these instructions. The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.	0b0
[24]	FZ	Flushing denormalized numbers to zero control bit. **0b0** Flushing denormalized numbers to zero disabled. **0b1** Flushing denormalized numbers to zero enabled. The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.	0b0
[23:22]	RMode	Rounding Mode control field. **0b00** Round to Nearest (RN) mode. **0b01** Round towards Plus Infinity (RP) mode. **0b10** Round towards Minus Infinity (RM) mode. **0b11** Round towards Zero (RZ) mode. The specified rounding mode is used by both scalar and Advanced SIMD floating-point instructions.	0b00
[21:20]	Stride	This field has no function in AArch64 state, and non-zero values are ignored during execution in AArch64 state.	0b00
[19]	FZ16	Flushing denormalized numbers to zero control bit on half-precision data-processing instructions. **0b0** For some instructions, this bit disables flushing to zero of inputs and outputs that are half-precision denormalized numbers. For more information, see 'Flushing denormalized numbers to zero'. **0b1** Flushing denormalized numbers to zero enabled. The value of this bit applies to both scalar and Advanced SIMD floating-point half-precision calculations.	0b0
[18:16]	Len	This field has no function in AArch64 state, and non-zero values are ignored during execution in AArch64 state.	0b000
[15:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, FPCR

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b000

MSR FPCR, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b000

Accessibility

MRS <Xt>, FPCR

```
if PSTATE.EL == EL0 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.FPEN != '11'
 then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        return FPCR;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif CPACR_EL1.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        return FPCR;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        return FPCR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        return FPCR;
```

MSR FPCR, <Xt>

```
if PSTATE.EL == EL0 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.FPEN != '11'
 then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t];
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif CPACR_EL1.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t];
```

A.1.44 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-60: AArch64_afsr0_el2 bit assignments

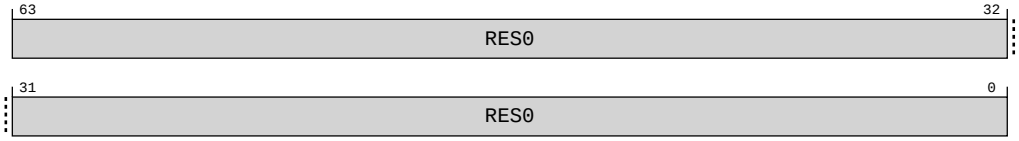


Table A-140: AFSR0_EL2 bit descriptions

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);
    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);
    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.1.45 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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-61: AArch64_afsr1_el2 bit assignments

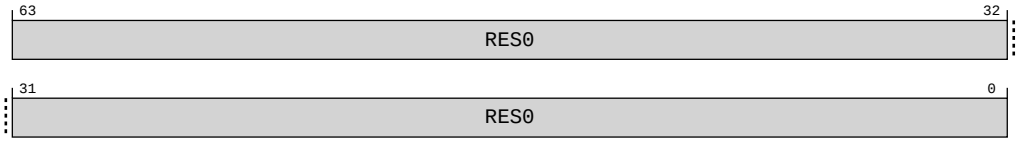


Table A-145: AFSR1_EL2 bit descriptions

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);
    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);
    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.1.46 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-62: AArch64_afsr0_el1 bit assignments

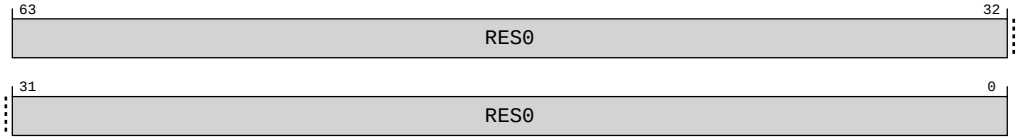


Table A-150: AFSR0_EL1 bit descriptions

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);
    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);
    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.1.47 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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-63: AArch64_afsr1_el1 bit assignments

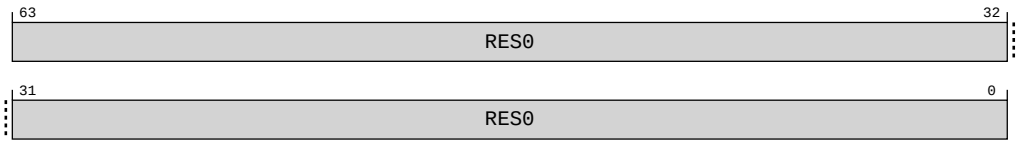


Table A-155: AFSR1_EL1 bit descriptions

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);
    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);
    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.1.48 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-64: AArch64_afsr0_el3 bit assignments

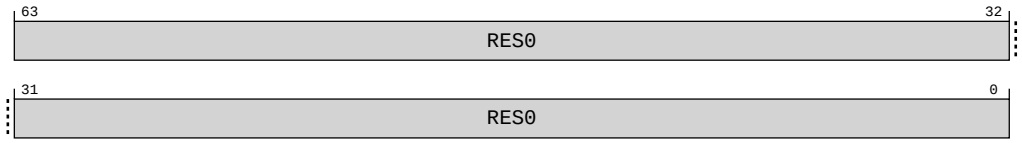


Table A-160: AFSR0_EL3 bit descriptions

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.1.49 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-65: AArch64_afsr1_el3 bit assignments

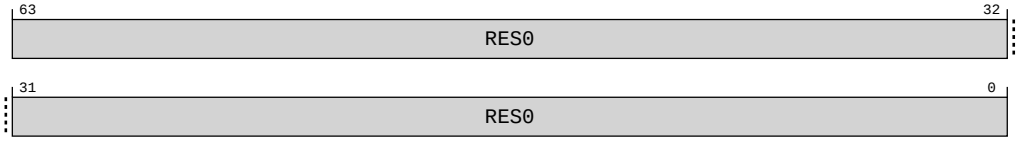


Table A-163: AFSR1_EL3 bit descriptions

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.2 AArch64 Special-purpose registers summary

The summary table provides an overview of **IMPLEMENTATION DEFINED** Special-purpose registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table A-166: Special-purpose registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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 468.

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 xx00


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-66: AArch64_imp_cpuppmcr_el3 bit assignments

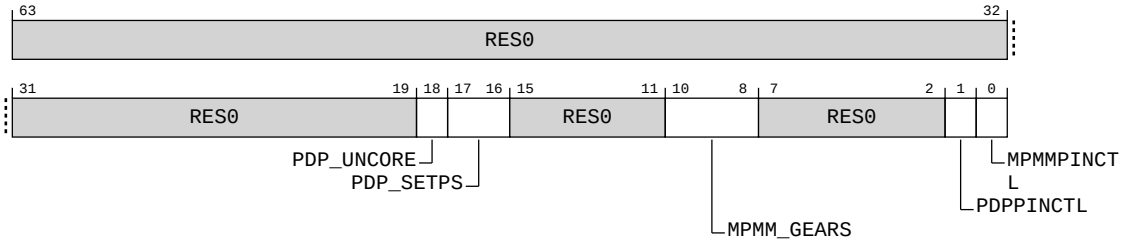


Table A-167: IMP_CPUPPMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18]	PDP_UNCORE	Indicates whether PDP uncore is implemented **0b1** PDP has separate uncore and core controls. Access to this field is: RO	x
[17:16]	PDP_SETPS	Number of PDP Setpoints implemented **0b11** 3 PDP are 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
[7:2]	**RES0**	Reserved	**RES0**
[1]	PDPPINCTL	PDP Pin Control Enabled **0b0** PDP control through SPR and utility bus **0b1** PDP control through pin only.	0b0
[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

A.3 AArch64 This register is a system instruction registers summary

The summary table provides an overview of **IMPLEMENTATION DEFINED** This register is a system instruction registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table A-170: This register is a system instruction registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
RAMINDEX	1	6	C15	C0	0	—	64-bit	RAMINDEX system instruction

A.3.1 RAMINDEX, RAMINDEX system instruction

.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

This register is a system instruction

Access type

See bit descriptions

Bit descriptions

When AArch64-RAMINDEX.ID == 0x0 and 64KB

Figure A-67: AArch64_ramindex bit assignments

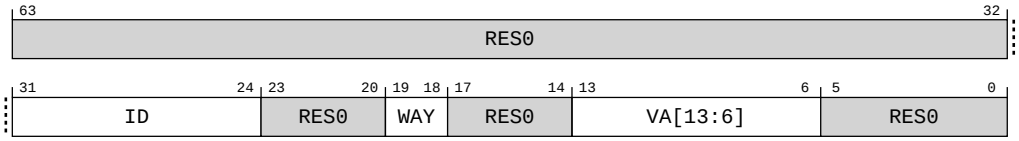


Table A-171: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	ID	ID of the selected memory **0b00000000** L1_I TAG	8{x}
[23:20]	**RES0**	Reserved	**RES0**
[19:18]	WAY	Way	xx
[17:14]	**RES0**	Reserved	**RES0**
[13:6]	VA[13:6]	Virtual Address bits[13:6]	8{x}
[5:0]	**RES0**	Reserved	**RES0**

When AArch64-RAMINDEX.ID == 0x0 and 32KB

Figure A-68: AArch64_ramindex bit assignments

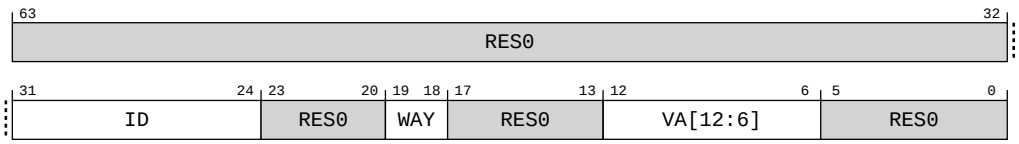


Table A-172: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	ID	ID of the selected memory **0b00000000** L1_I TAG	8{x}
[23:20]	**RES0**	Reserved	**RES0**
[19:18]	WAY	Way	xx
[17:13]	**RES0**	Reserved	**RES0**
[12:6]	VA[12:6]	Virtual Address bits[12:6]	7{x}
[5:0]	**RES0**	Reserved	**RES0**

When AArch64-RAMINDEX.ID == 0x1 and 64KB

Figure A-69: AArch64_ramindex bit assignments

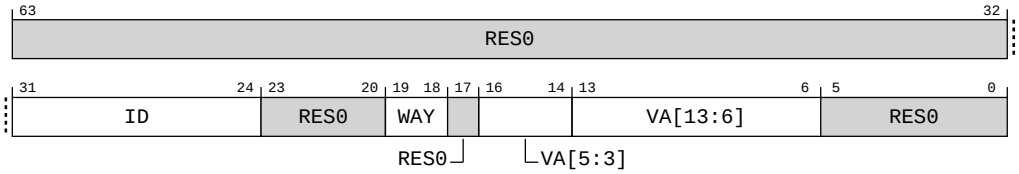


Table A-173: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	ID	ID of the selected memory **0b00000001** L1_I Data	8{x}
[23:20]	**RES0**	Reserved	**RES0**
[19:18]	WAY	Way	xx
[17]	**RES0**	Reserved	**RES0**
[16:14]	VA[5:3]	Virtual Address bits[5:3]	xxx
[13:6]	VA[13:6]	Virtual Address bits[13:6]	8{x}
[5:0]	**RES0**	Reserved	**RES0**

When AArch64-RAMINDEX.ID == 0x1 and 32KB

Figure A-70: AArch64_ramindex bit assignments

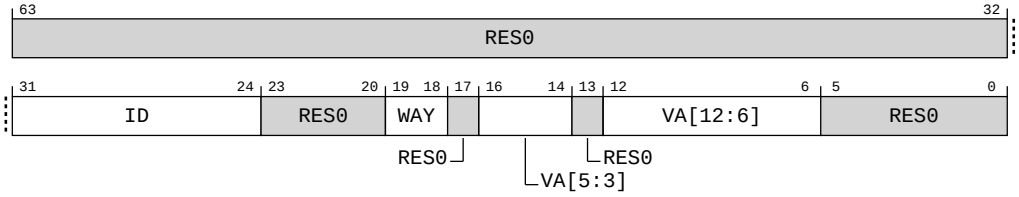


Table A-174: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	ID	ID of the selected memory **0b00000001** L1_I Data	8{x}
[23:20]	**RES0**	Reserved	**RES0**
[19:18]	WAY	Way	xx
[17]	**RES0**	Reserved	**RES0**
[16:14]	VA[5:3]	Virtual Address bits[5:3]	xxx
[13]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[12:6]	VA[12:6]	Virtual Address bits[12:6]	7{x}
[5:0]	RES0	Reserved	RES0

When AArch64-RAMINDEX.ID == 0x8 and 64KB

Figure A-71: AArch64_ramindex bit assignments

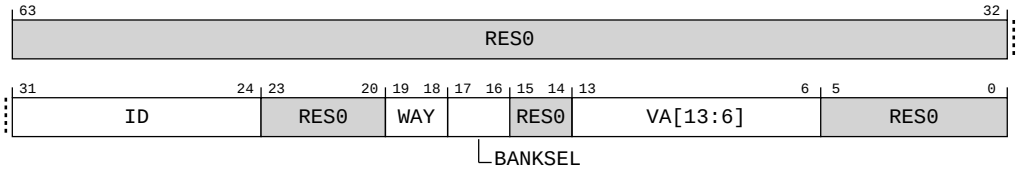


Table A-175: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:24]	ID	ID of the selected memory **0b00001000** L1_D Tag	8{x}
[23:20]	RES0	Reserved	RES0
[19:18]	WAY	Way	xx
[17:16]	BANKSEL	Bank selection **0b00** Tag RAM 0 **0b01** Tag RAM 1 **0b10** Tag RAM 2	xx
[15:14]	RES0	Reserved	RES0
[13:6]	VA[13:6]	Virtual Address bits[13:6]	8{x}
[5:0]	RES0	Reserved	RES0

When AArch64-RAMINDEX.ID == 0x8 and 32KB

Figure A-72: AArch64_ramindex bit assignments

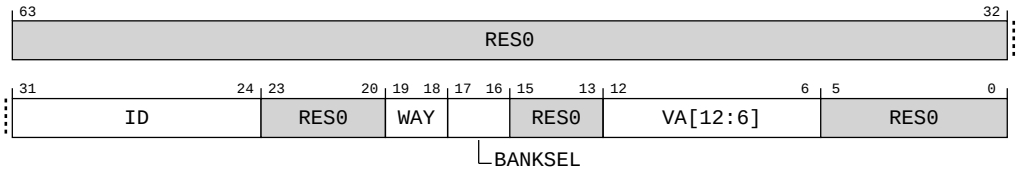


Table A-176: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	ID	ID of the selected memory **0b00001000** L1_D Tag	8{x}
[23:20]	**RES0**	Reserved	**RES0**
[19:18]	WAY	Way	xx
[17:16]	BANKSEL	Bank selection **0b00** Tag RAM 0 **0b01** Tag RAM 1 **0b10** Tag RAM 2	xx
[15:13]	**RES0**	Reserved	**RES0**
[12:6]	VA[12:6]	Virtual Address bits[12:6]	7{x}
[5:0]	**RES0**	Reserved	**RES0**

When AArch64-RAMINDEX.ID == 0x9 and 64KB

Figure A-73: AArch64_ramindex bit assignments

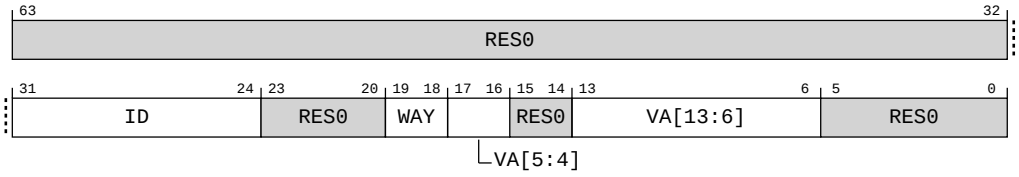


Table A-177: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	ID	ID of the selected memory **0b00001001** L1_D Data	8{x}
[23:20]	**RES0**	Reserved	**RES0**
[19:18]	WAY	Way	xx
[17:16]	VA[5:4]	Virtual Address bits[5:4]	xx
[15:14]	**RES0**	Reserved	**RES0**
[13:6]	VA[13:6]	Virtual Address bits[13:6]	8{x}
[5:0]	**RES0**	Reserved	**RES0**

When AArch64-RAMINDEX.ID == 0x9 and 32KB

Figure A-74: AArch64_ramindex bit assignments

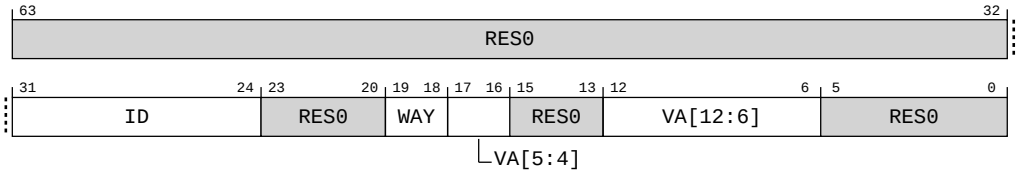


Table A-178: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	ID	ID of the selected memory **0b00001001** L1_D Data	8{x}
[23:20]	**RES0**	Reserved	**RES0**
[19:18]	WAY	Way	xx
[17:16]	VA[5:4]	Virtual Address bits[5:4]	xx
[15:13]	**RES0**	Reserved	**RES0**
[12:6]	VA[12:6]	Virtual Address bits[12:6]	7{x}
[5:0]	**RES0**	Reserved	**RES0**

When AArch64-RAMINDEX.ID == 0x10 or AArch64-RAMINDEX.ID == 0x11 and 512KB

Figure A-75: AArch64_ramindex bit assignments

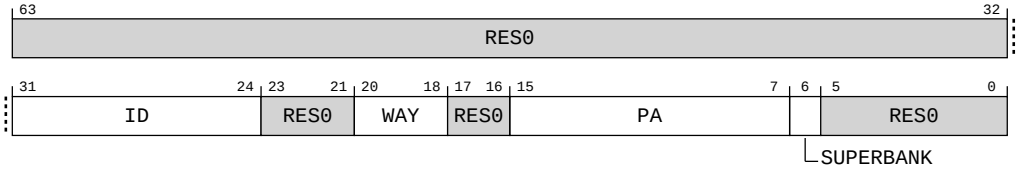


Table A-179: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	ID	ID of the selected memory **0b00010000** L2 Tag **0b00010001** L2 Data	8{x}
[23:21]	**RES0**	Reserved	**RES0**
[20:18]	WAY	Way	xxx
[17:16]	**RES0**	Reserved	**RES0**
[15:7]	PA	Physical Address bits[15:7]	9{x}

Bits	Name	Description	Reset
[6]	SUPERBANK	Physical Address bit[6]	x
[5:0]	RES0	Reserved	RES0

When AArch64-RAMINDEX.ID == 0x10 or AArch64-RAMINDEX.ID == 0x11 and 256KB

Figure A-76: AArch64_ramindex bit assignments

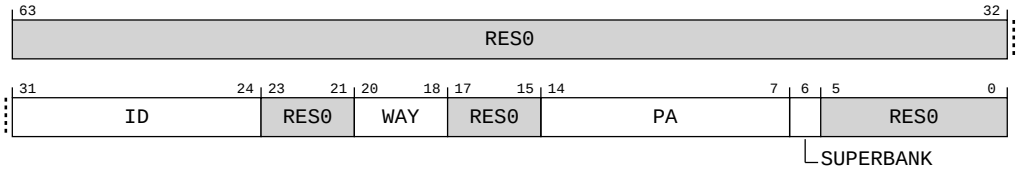


Table A-180: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:24]	ID	ID of the selected memory **0b00010000** L2 Tag **0b00010001** L2 Data	8{x}
[23:21]	RES0	Reserved	RES0
[20:18]	WAY	Way	xxx
[17:15]	RES0	Reserved	RES0
[14:7]	PA	Physical Address bits[14:7]	8{x}
[6]	SUPERBANK	Physical Address bit[6]	x
[5:0]	RES0	Reserved	RES0

When AArch64-RAMINDEX.ID == 0x10 or AArch64-RAMINDEX.ID == 0x11 and 128KB

Figure A-77: AArch64_ramindex bit assignments

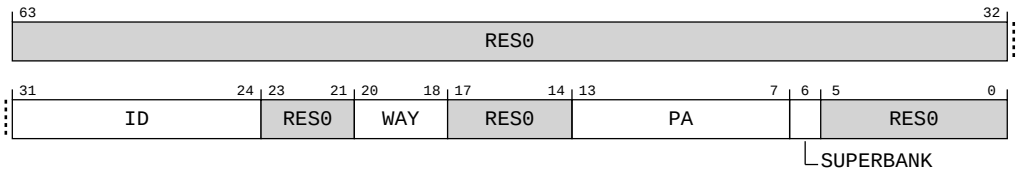


Table A-181: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:24]	ID	ID of the selected memory **0b00010000** L2 Tag **0b00010001** L2 Data	8{x}
[23:21]	**RES0**	Reserved	**RES0**
[20:18]	WAY	Way	xxx
[17:14]	**RES0**	Reserved	**RES0**
[13:7]	PA	Physical Address bits[13:7]	7{x}
[6]	SUPERBANK	Physical Address bit[6]	x
[5:0]	**RES0**	Reserved	**RES0**

When AArch64-RAMINDEX.ID == 0x18

Figure A-78: AArch64_ramindex bit assignments

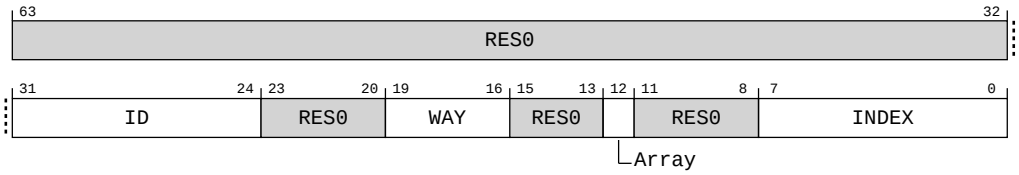


Table A-182: RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	ID	ID of the selected memory **0b00011000** TLB	8{x}
[23:20]	**RES0**	Reserved	**RES0**
[19:16]	WAY	Way	xxxx
[15:13]	**RES0**	Reserved	**RES0**
[12]	Array	Array **0b0** TCSP **0b1** TCMP	x
[11:8]	**RES0**	Reserved	**RES0**
[7:0]	INDEX	Index	8{x}

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C0, #0, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0000	0b000

Accessibility

```
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
    RAMINDEX(X[t]);
```

A.4 AArch64 Identification registers summary

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table A-184: 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
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_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
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_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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ID_AA64MMFR2_EL1	3	0	C0	C7	2	–	64-bit	AArch64 Memory Model Feature Register 2
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
MPAMIDR_EL1	3	0	C10	C4	4	–	64-bit	MPAM ID Register (EL1)
IMP_CPUPPMPDPCR_EL1	3	0	C15	C2	4	–	64-bit	Global PPMPDP Configuration Register
IMP_CPUMPMMCR_EL3	3	6	C15	C2	1	–	64-bit	Global MPMM Configuration Register

A.4.1 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.3.4 MIDR_EL1, External Main ID Register on page 547.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0100 0001 0001 1111 1101 0100 1101 0010

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-79: AArch64_midr_el1 bit assignments

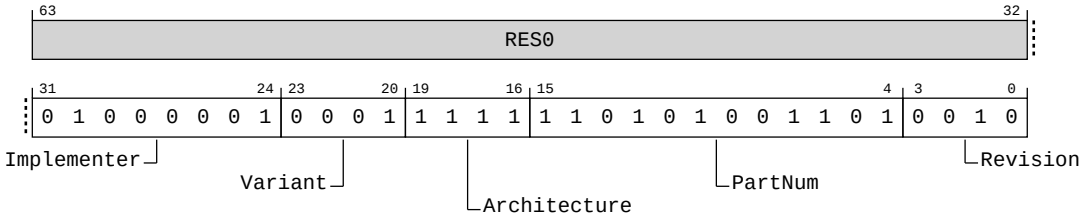


Table A-185: MIDR_EL1 bit descriptions

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. **0b0001** r1p2	0b0001
[19:16]	Architecture	Architecture version. For A-profile, the 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. **0b110101001101** A715	0xD4D
[3:0]	Revision	Indicates the minor revision of the product. **0b0010** r1p2	0b0010

Access

MRS <Xt>, MIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b000

A.4.2 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-80: AArch64_mpidr_el1 bit assignments

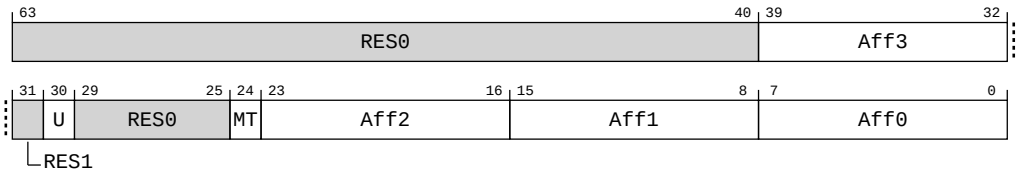


Table A-187: MPIDR_EL1 bit descriptions

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**

Bits	Name	Description	Reset
[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. A715 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() 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.4.3 REVIDR_EL1, Revision ID Register

Provides implementation-specific minor revision information.

Configurations

If REVIDR_EL1 has the same value as AArch64-MIDR_EL1, then its contents have no significance.

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-81: AArch64_revidr_el1 bit assignments

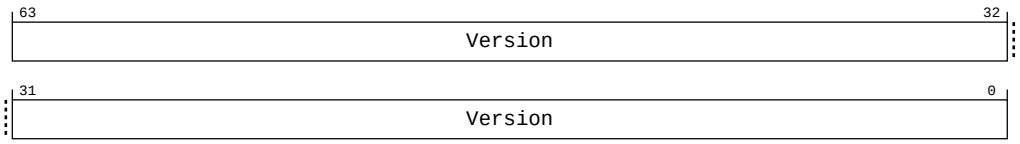


Table A-189: REVIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Version	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);
    else
        return REVIDR_EL1;
elsif PSTATE.EL == EL2 then
    return REVIDR_EL1;
elsif PSTATE.EL == EL3 then
    return REVIDR_EL1;
```

A.4.4 MVFR0_EL1, AArch32 Media and VFP Feature Register 0

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with AArch64-MVFR1_EL1 and AArch64-MVFR2_EL1.

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

In an implementation where at least one Exception level supports execution in AArch32 state, but there is no support for Advanced SIMD and floating-point operation, this register is RAZ.

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-82: AArch64_mvfr0_el1 bit assignments

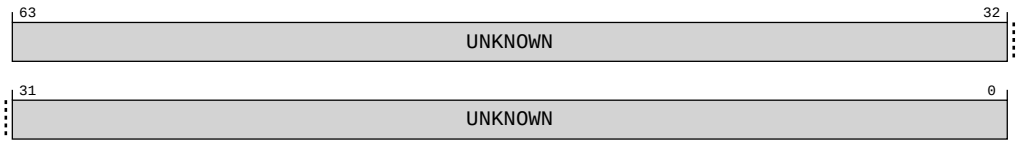


Table A-191: MVFR0_EL1 bit descriptions

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

Access

MRS <Xt>, MVFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	0b000

Accessibility

MRS <Xt>, MVFR0_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 MVFR0_EL1;
elsif PSTATE.EL == EL2 then
    return MVFR0_EL1;
elsif PSTATE.EL == EL3 then
    return MVFR0_EL1;
```

A.4.5 MVFR1_EL1, AArch32 Media and VFP Feature Register 1

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with AArch64-MVFR0_EL1 and AArch64-MVFR2_EL1.

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

In an implementation where at least one Exception level supports execution in AArch32 state, but there is no support for Advanced SIMD and floating-point operation, this register is RAZ.

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-83: AArch64_mvfr1_el1 bit assignments

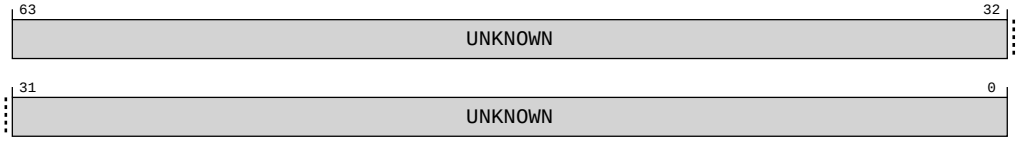


Table A-193: MVFR1_EL1 bit descriptions

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

Access

MRS <Xt>, MVFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	0b001

Accessibility

MRS <Xt>, MVFR1_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 MVFR1_EL1;
elsif PSTATE.EL == EL2 then
    return MVFR1_EL1;
elsif PSTATE.EL == EL3 then
    return MVFR1_EL1;
```

A.4.6 MVFR2_EL1, AArch32 Media and VFP Feature Register 2

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with AArch64-MVFR0_EL1 and AArch64-MVFR1_EL1.

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

In an implementation where at least one Exception level supports execution in AArch32 state, but there is no support for Advanced SIMD and floating-point operation, this register is RAZ.

Attributes

Width

64

Functional group

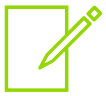
Identification registers

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-84: AArch64_mvfr2_el1 bit assignments

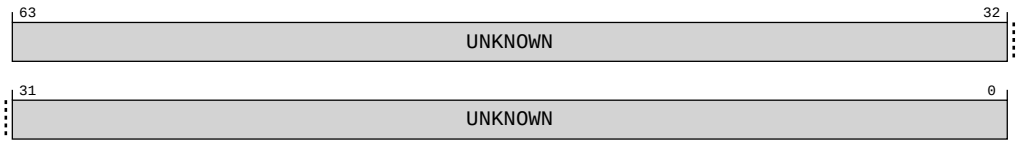


Table A-195: MVFR2_EL1 bit descriptions

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

Access

MRS <Xt>, MVFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	0b010

Accessibility

MRS <Xt>, MVFR2_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 MVFR2_EL1;
elsif PSTATE.EL == EL2 then
    return MVFR2_EL1;
elsif PSTATE.EL == EL3 then
    return MVFR2_EL1;
```

A.4.7 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

0001 0010 xxxx 0001 0001 0001 0001 0001 0010 xxxx 0001 0001 0001 0001 0001 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-85: AArch64_id_aa64pfr0_el1 bit assignments

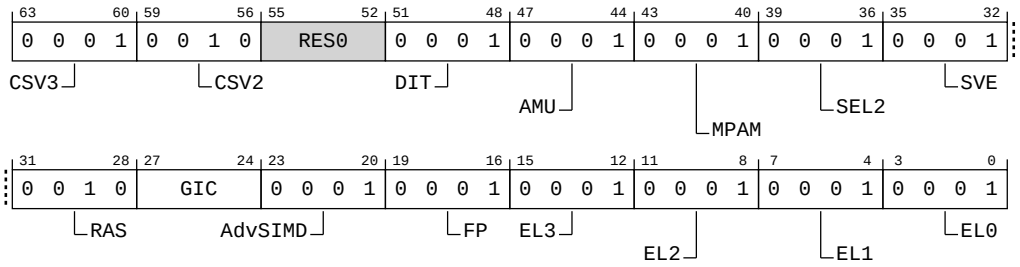


Table A-197: ID_AA64PFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	CSV3	Speculative use of faulting data. Defined values are: **0b0001** Data loaded under speculation with a permission or domain fault cannot be used to form an address or generate condition codes or SVE predicate values to be used by instructions newer than the load in the speculative sequence.	0b0001
[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.	0b0010
[55:52]	**RES0**	Reserved	**RES0**

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.	0b0001
[47:44]	AMU	Indicates support for Activity Monitors Extension. Defined values are: **0b0001** FEAT_AMUv1 is implemented.	0b0001
[43:40]	MPAM	Indicates support for MPAM Extension. Defined values are: **0b0001** If AArch64-ID_AA64PFR1_EL1.MPAM_frac == 0b0000, MPAM Extension version 1.0 is implemented. If AArch64-ID_AA64PFR1_EL1.MPAM_frac == 0b0001, MPAM Extension version 1.1 is implemented.	0b0001
[39:36]	SEL2	Secure EL2. Defined values are: **0b0001** Secure EL2 is implemented.	0b0001
[35:32]	SVE	Scalable Vector Extension. Defined values are: **0b0001** SVE architectural state and programmers' model are implemented.	0b0001
[31:28]	RAS	RAS Extension version. Defined values are: **0b0010** FEAT_RASv1p1 is implemented.	0b0010
[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.	The reset values can be the following: 0b0000, 0b0011, respective to the value.
[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.	0b0001

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.	0b0001
[15:12]	EL3	EL3 Exception level handling. Defined values are: **0b0001** EL3 can be executed in AArch64 state only.	0b0001
[11:8]	EL2	EL2 Exception level handling. Defined values are: **0b0001** EL2 can be executed in AArch64 state only.	0b0001
[7:4]	EL1	EL1 Exception level handling. Defined values are: **0b0001** EL1 can be executed in AArch64 state only.	0b0001
[3:0]	EL0	EL0 Exception level handling. Defined values are: **0b0001** EL0 can be executed in AArch64 state only.	0b0001

Access

MRS <Xt>, ID_AA64PFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b000

Accessibility

MRS <Xt>, ID_AA64PFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64PFR0_EL1;
```

A.4.8 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 0000 xxxx xxxx xxxx xxxx xxxx xxxx 0010 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-86: AArch64_id_aa64pfr1_el1 bit assignments

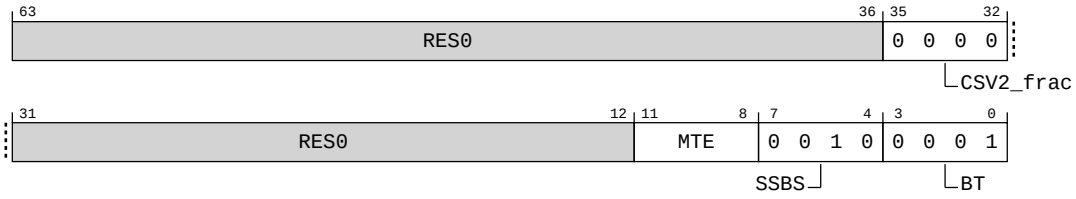


Table A-199: ID_AA64PFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:36]	RES0	Reserved	RES0
[35:32]	CSV2_frac	CSV2 fractional field. Defined values are: **0b0000** This PE does not disclose whether branch targets trained in one hardware-described context can exploitatively control speculative execution in a different hardware-described context. The SCXTNUM_ELx registers are not supported.	0b0000

Bits	Name	Description	Reset
[31:12]	**RES0**	Reserved	**RES0**
[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.	The reset values can be the following: 0b0001, 0b0011, respective to the value.
[7:4]	SSBS	Speculative Store Bypassing controls in AArch64 state. Defined values are: **0b0010** AArch64 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypassing Safe, and the MSR and MRS instructions to directly read and write the PSTATE.SSBS field.	0b0010
[3:0]	BT	Branch Target Identification mechanism support in AArch64 state. Defined values are: **0b0001** The Branch Target Identification mechanism is implemented.	0b0001

Access

MRS <Xt>, ID_AA64PFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b001

Accessibility

MRS <Xt>, ID_AA64PFR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64PFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64PFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64PFR1_EL1;
```

A.4.9 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 0001 xxxx xxxx xxxx xxxx xxxx 0001 0001 xxxx xxxx xxxx 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-87: AArch64_id_aa64zfr0_el1 bit assignments

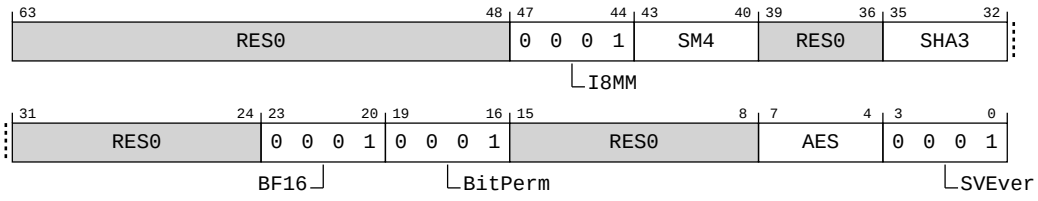


Table A-201: ID_AA64ZFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**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.	0b0001
[43:40]	SM4	Indicates support for SVE SM4 instructions. Defined values are: **0b0000** SVE2 SM4 instructions are not implemented. This value is reported when the Cryptographic Extension is not implemented or is disabled. **0b0001** SVE2 SM4E and SM4EKEY instructions are implemented. This value is reported when the Cryptographic Extension is implemented and enabled.	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[39:36]	**RES0**	Reserved	**RES0**
[35:32]	SHA3	Indicates support for the SVE SHA3 instructions. Defined values are: **0b0000** SVE2 SHA3 instructions are not implemented. This value is reported when the Cryptographic Extension is not implemented or is disabled. **0b0001** SVE2 RAX1 instruction is implemented. This value is reported when the Cryptographic Extension is implemented and enabled.	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[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.	0b0001
[19:16]	BitPerm	Indicates support for SVE bit permute instructions. Defined values are: **0b0001** SVE BDEP, BEXT, and BGRP instructions are implemented.	0b0001
[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 the Cryptographic Extension is not implemented or is disabled. **0b0010** SVE2 AESE, AESD, AESMC, and AESIMC instructions are implemented plus SVE2 PMULLB and PMULLT instructions with 64-bit source. This value is reported when the Cryptographic Extension is implemented and enabled.	The reset values can be the following: 0b0000, 0b0010, respective to the value.
[3:0]	SVEver	Indicates support for SVE. Defined values are: **0b0001** SVE and the non-optional SVE2 instructions are implemented.	0b0001

Access

MRS <Xt>, ID_AA64ZFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b100

Accessibility

MRS <Xt>, ID_AA64ZFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ZFR0_EL1;
```

A.4.10 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 0001 0001 1111 xxxx 0001 xxxx 0011 xxxx 0101 0110 0001 1001


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-88: AArch64_id_aa64dfr0_el1 bit assignments

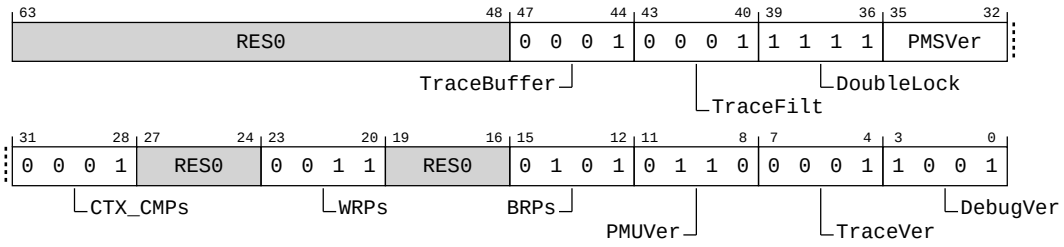


Table A-203: ID_AA64DFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:44]	TraceBuffer	Trace Buffer Extension. Defined values are: **0b0001** Trace Buffer Extension implemented, FEAT_TRBE.	0b0001
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. Defined values are: **0b0001** Armv8.4 Self-hosted Trace Extension implemented.	0b0001
[39:36]	DoubleLock	OS Double Lock implemented. Defined values are: **0b1111** OS Double Lock not implemented. AArch64-OSDLR_EL1 is **RAZ/WI**.	0b1111
[35:32]	PMSVer	Statistical Profiling Extension version. Defined values are: **0b0000** Statistical Profiling Extension not implemented. **0b0010** Statistical Profiling Extension version SPEv1p1 is implemented.	The reset values can be the following: 0b0000, 0b0010, respective to the value.
[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	0b0001
[27:24]	**RES0**	Reserved	**RES0**
[23:20]	WRPs	Number of watchpoints, minus 1. The value of 0b0000 is reserved. **0b0011** Four watchpoints	0b0011
[19:16]	**RES0**	Reserved	**RES0**
[15:12]	BRPs	Number of breakpoints, minus 1. The value of 0b0000 is reserved. **0b0101** Six breakpoints	0b0101

Bits	Name	Description	Reset
[11:8]	PMUVer	Performance Monitors Extension version. Defined value is: **0b0110** Performance Monitors Extension implemented, PMUv3 for Armv8.5	0b0110
[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.	0b0001
[3:0]	DebugVer	Debug architecture version. Indicates presence of Armv8 debug architecture. Defined values are: **0b1001** Armv8.4 debug architecture.	0b1001

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.4.11 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-89: AArch64_id_aa64dfr1_el1 bit assignments

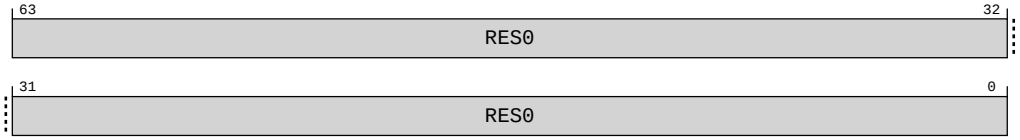


Table A-205: ID_AA64DFR1_EL1 bit descriptions

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.4.12 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-90: AArch64_id_aa64afr0_el1 bit assignments

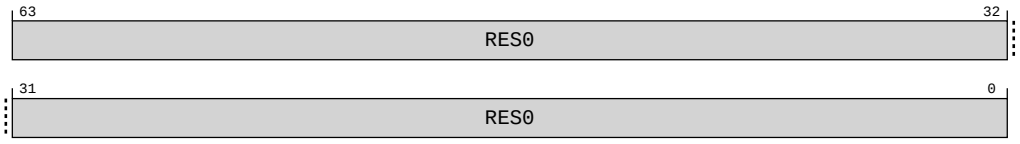


Table A-207: ID_AA64AFR0_EL1 bit descriptions

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.4.13 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-91: AArch64_id_aa64afr1_el1 bit assignments

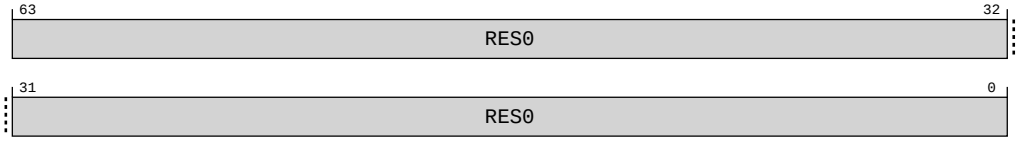


Table A-209: ID_AA64AFR1_EL1 bit descriptions

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.4.14 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

0000 0010 0010 0001 0001 xxxx xxxx xxxx 0001 0000 0010 0001 xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-92: AArch64_id_aa64isar0_el1 bit assignments

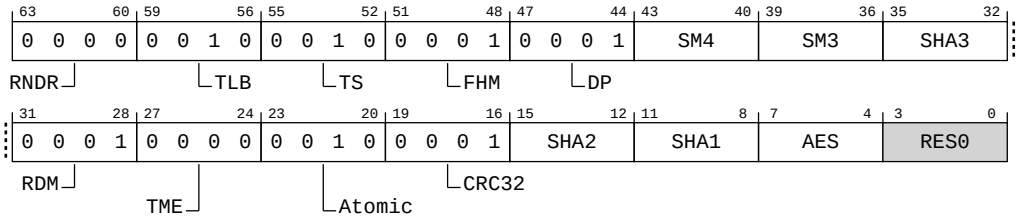


Table A-211: ID_AA64ISAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	RNDR	Indicates support for Random Number instructions in AArch64 state. Defined values are: **0b0000** No Random Number instructions are implemented.	0b0000
[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.	0b0010

Bits	Name	Description	Reset
[55:52]	TS	Indicates support for flag manipulation instructions. Defined values are: **0b0010** CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented.	0b0010
[51:48]	FHM	Indicates support for FMLAL and FMLSL instructions. Defined values are: **0b0001** FMLAL and FMLSL instructions are implemented.	0b0001
[47:44]	DP	Indicates support for Dot Product instructions in AArch64 state. Defined values are: **0b0001** UDOT and SDOT instructions implemented.	0b0001
[43:40]	SM4	Indicates support for SM4 instructions in AArch64 state. Defined values are: **0b0000** No SM4 instructions implemented. This value is reported when the Cryptographic Extension is not implemented or is disabled. **0b0001** SM4E and SM4EKEY instructions implemented. This value is reported when the Cryptographic Extension is implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset the Cryptographic Extension is implemented	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[39:36]	SM3	Indicates support for SM3 instructions in AArch64 state. Defined values are: **0b0000** No SM3 instructions implemented. This value is reported when the Cryptographic Extension is not implemented or is disabled. **0b0001** SM3SS1, SM3TT1A, SM3TT1B, SM3TT2A, SM3TT2B, SM3PARTW1, and SM3PARTW2 instructions implemented. This value is reported when the Cryptographic Extension is implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset the Cryptographic Extension is implemented	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[35:32]	SHA3	Indicates support for SHA3 instructions in AArch64 state. Defined values are: **0b0000** No SHA3 instructions implemented. This value is reported when the Cryptographic Extension is not implemented or is disabled. **0b0001** EOR3, RAX1, XAR, and BCAX instructions implemented. This value is reported when the Cryptographic Extension is implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset the Cryptographic Extension is implemented	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[31:28]	RDM	Indicates support for SQRDMLAH and SQRDMLSH instructions in AArch64 state. Defined values are: **0b0001** SQRDMLAH and SQRDMLSH instructions implemented.	0b0001

Bits	Name	Description	Reset
[27:24]	TME	Indicates support for TME instructions. Defined values are: **0b0000** TME instructions are not implemented.	0b0000
[23:20]	Atomic	Indicates support for Atomic instructions in AArch64 state. Defined values are: **0b0010** LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions implemented.	0b0010
[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.	0b0001
[15:12]	SHA2	Indicates support for SHA2 instructions in AArch64 state. Defined values are: **0b0000** No SHA2 instructions implemented. This value is reported when the Cryptographic Extension is not implemented or is disabled. **0b0010** SHA256H, SHA256H2, SHA256SU0, SHA256SU1, SHA512H, SHA512H2, SHA512SU0, and SHA512SU1 instructions implemented. This value is reported when the Cryptographic Extension is implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset the Cryptographic Extension is implemented	The reset values can be the following: 0b0000, 0b0010, respective to the value.
[11:8]	SHA1	Indicates support for SHA1 instructions in AArch64 state. Defined values are: **0b0000** No SHA1 instructions implemented. This value is reported when the Cryptographic Extension is not implemented or is disabled. **0b0001** SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions implemented. This value is reported when the Cryptographic Extension is implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset the Cryptographic Extension is implemented	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[7:4]	AES	Indicates support for AES instructions in AArch64 state. Defined values are: **0b0000** No AES instructions implemented. This value is reported when the Cryptographic Extension is not implemented or is 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 the Cryptographic Extension is implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset the Cryptographic Extension is implemented	The reset values can be the following: 0b0000, 0b0010, respective to the value.
[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.4.15 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

If ID_AA64ISAR1_EL1.{API, APA} == {0000, 0000}, then:

- The AArch64-TCR_EL1.{TBID,TBID0}, AArch64-TCR_EL2.{TBID0,TBID1}, AArch64-TCR_EL2.TBID and AArch64-TCR_EL3.TBID bits are RES0.
- AArch64-APIAKeyHi_EL1, AArch64-APIAKeyLo_EL1, AArch64-APIBKeyHi_EL1, AArch64-APIBKeyLo_EL1, AArch64-APDAKeyHi_EL1, AArch64-APDAKeyLo_EL1, AArch64-APDBKeyHi_EL1, AArch64-APDBKeyLo_EL1 are not allocated.
- SCTLR_ELx.EnIA, SCTLR_ELx.EnIB, SCTLR_ELx.EnDA, SCTLR_ELx.EnDB are all RES0.

If ID_AA64ISAR1_EL1.{GPI, GPA, API, APA} == {0000, 0000, 0000, 0000}, then:

- AArch64-HCR_EL2.APK and AArch64-HCR_EL2.API are RES0.
- AArch64-SCR_EL3.APK and AArch64-SCR_EL3.API are RES0.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx 0001 0000 0001 0001 0001 0001 0000 0001 0010 0001 0001 0000 0101 0010

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-93: AArch64_id_aa64isar1_el1 bit assignments

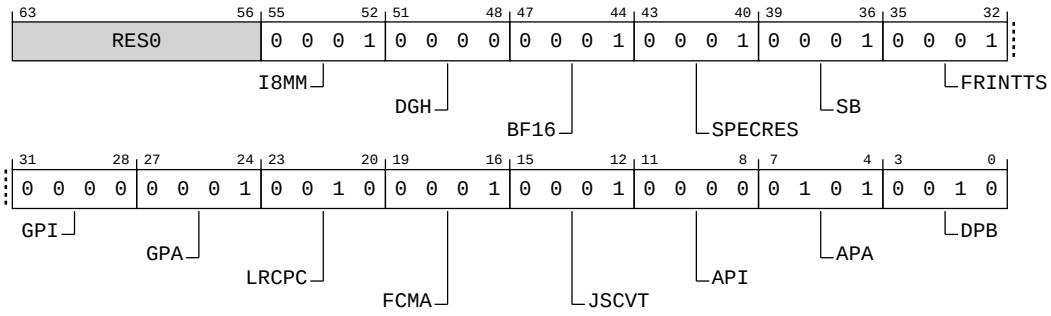


Table A-213: ID_AA64ISAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[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.	0b0001
[51:48]	DGH	Indicates support for the Data Gathering Hint instruction. Defined values are: **0b0000** Data Gathering Hint is not implemented.	0b0000
[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.	0b0001

Bits	Name	Description	Reset
[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.	0b0001
[39:36]	SB	Indicates support for SB instruction in AArch64 state. Defined values are: **0b0001** SB instruction is implemented.	0b0001
[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.	0b0001
[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.	0b0000
[27:24]	GPA	Indicates whether QARMA5 or Architected algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: **0b0001** Generic Authentication using the QARMA5 algorithm is implemented. This includes the PACGA instruction.	0b0001
[23:20]	LRCPC	Indicates support for weaker release consistency, RCpc, based model. Defined values are: **0b0010** The LDAPUR*, STLUR*, and LDAPR* instructions are implemented.	0b0010
[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.	0b0001
[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.	0b0001
[11:8]	API	Indicates whether an **IMPLEMENTATION DEFINED** algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are: **0b0000** Address Authentication using an **IMPLEMENTATION DEFINED** algorithm is not implemented.	0b0000
[7:4]	APA	Indicates whether QARMA5 or Architected algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are: **0b0101** Address Authentication using the QARMA5 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.	0b0101

Bits	Name	Description	Reset
[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	0b0010

Access

MRS <Xt>, ID_AA64ISAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b001

Accessibility

MRS <Xt>, ID_AA64ISAR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ISAR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ISAR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ISAR1_EL1;
```

A.4.16 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx 0000 0010 0010 0010 0000 0000 0001 0000 0001 0001 0010 0010

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-94: AArch64_id_aa64mmfr0_el1 bit assignments

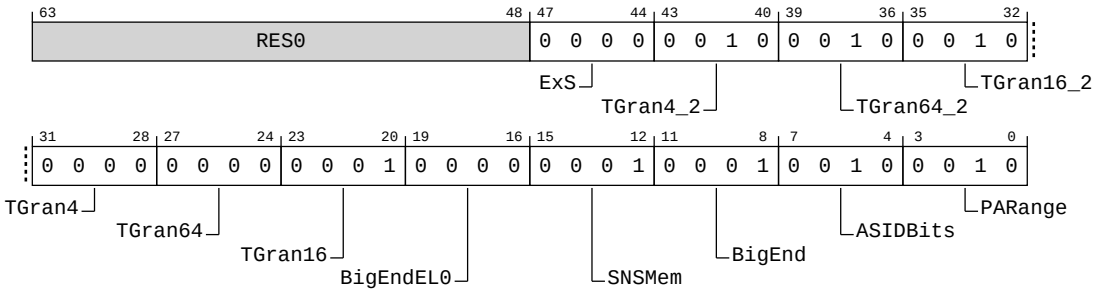


Table A-215: ID_AA64MMFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**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.	0b0000
[43:40]	TGran4_2	Indicates support for 4KB memory granule size at stage 2. Defined values are: **0b0010** 4KB granule supported at stage 2.	0b0010
[39:36]	TGran64_2	Indicates support for 64KB memory granule size at stage 2. Defined values are: **0b0010** 64KB granule supported at stage 2.	0b0010
[35:32]	TGran16_2	Indicates support for 16KB memory granule size at stage 2. Defined values are: **0b0010** 16KB granule supported at stage 2.	0b0010
[31:28]	TGran4	Indicates support for 4KB memory translation granule size. Defined values are: **0b0000** 4KB granule supported.	0b0000
[27:24]	TGran64	Indicates support for 64KB memory translation granule size. Defined values are: **0b0000** 64KB granule supported.	0b0000

Bits	Name	Description	Reset
[23:20]	TGran16	Indicates support for 16KB memory translation granule size. Defined values are: **0b0001** 16KB granule supported.	0b0001
[19:16]	BigEndEL0	Indicates support for mixed-endian at EL0 only. Defined values are: **0b0000** No mixed-endian support at EL0. The AArch64-SCTLR_EL1.E0E bit has a fixed value.	0b0000
[15:12]	SNSMem	Indicates support for a distinction between Secure and Non-secure Memory. Defined values are: **0b0001** Does support a distinction between Secure and Non-secure Memory.	0b0001
[11:8]	BigEnd	Indicates support for mixed-endian configuration. Defined values are: **0b0001** Mixed-endian support. The *SCTLR_ELx* in the *Arm® Architecture Reference Manual for A-profile architecture*.EE and AArch64-SCTLR_EL1.E0E bits can be configured.	0b0001
[7:4]	ASIDBits	Number of ASID bits. Defined values are: **0b0010** 16 bits.	0b0010
[3:0]	PARange	Physical Address range supported. Defined values are: **0b0010** 40 bits, 1TB.	0b0010

Access

MRS <Xt>, ID_AA64MMFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b000

Accessibility

MRS <Xt>, ID_AA64MMFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64MMFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64MMFR0_EL1;
```

A.4.17 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0001 0000 0011 0001 0010 0001 0010 0010

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-95: AArch64_id_aa64mmfr1_el1 bit assignments

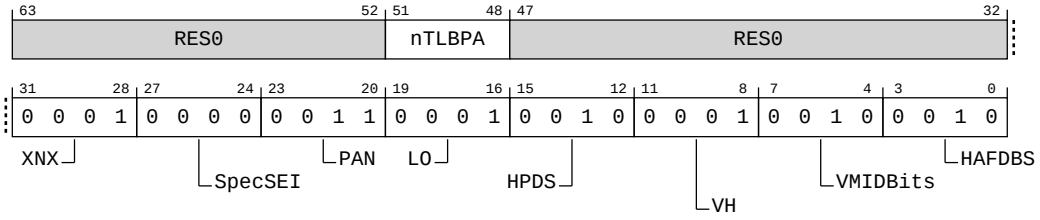


Table A-217: ID_AA64MMFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:52]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[51:48]	nTLBPA	Indicates support for intermediate caching of translation table walks. Defined values are: **0b0000** The intermediate caching of translation table walks might include non-coherent caches of previous valid translation table entries since the last completed relevant 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. **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. All other values are reserved. FEAT_nTLBPA implements the functionality identified by the value 0b0001. From Armv8.0, the permitted values are 0b0000 and 0b0001.	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[47:32]	**RES0**	Reserved	**RES0**
[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.	0b0001
[27:24]	SpecSEI	Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches. The defined values of this field are: **0b0000** The PE never generates an SError interrupt due to an External abort on a speculative read.	0b0000
[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.	0b0011
[19:16]	LO	LORegions. Indicates support for LORegions. Defined values are: **0b0001** LORegions supported.	0b0001
[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.	0b0010

Bits	Name	Description	Reset
[11:8]	VH	Virtualization Host Extensions. Defined values are: **0b0001** Virtualization Host Extensions supported.	0b0001
[7:4]	VMIDBits	Number of VMID bits. Defined values are: **0b0010** 16 bits	0b0010
[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.	0b0010

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.4.18 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

0001 0010 0010 0001 xxxx 0001 0001 0001 0001 0000 0001 0000 0001 0000 0001 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-96: AArch64_id_aa64mmfr2_el1 bit assignments

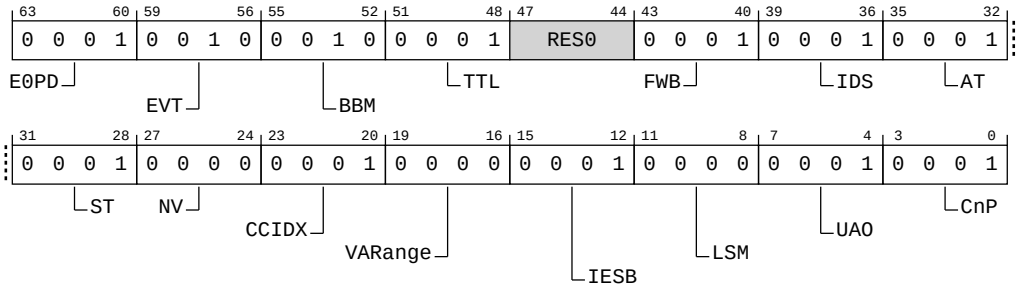


Table A-219: ID_AA64MMFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	E0PD	Indicates support for the E0PD mechanism. Defined values are: **0b0001** E0PDx mechanism is implemented.	0b0001

Bits	Name	Description	Reset
[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.	0b0010
[55:52]	BBM	Allows identification of the requirements of the hardware to have break-before-make sequences when changing block size for a translation. **0b0010** Level 2 support for changing block size is supported.	0b0010
[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.	0b0001
[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.	0b0001
[39:36]	IDS	Indicates the value of ESR_ELx.EC that reports an exception generated by a read access to the feature ID space. Defined values are: **0b0001** All exceptions generated by an AArch64 read access to the feature ID space are reported by ESR_ELx.EC == 0x18.	0b0001
[35:32]	AT	Identifies support for unaligned single-copy atomicity and atomic functions. Defined values are: **0b0001** Unaligned single-copy atomicity and atomic functions with a 16-byte address range aligned to 16-bytes are supported.	0b0001
[31:28]	ST	Identifies support for small translation tables. 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.	0b0001
[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.	0b0000
[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.	0b0001
[19:16]	VARange	Indicates support for a larger virtual address. Defined values are: **0b0000** VMSAv8-64 supports 48-bit VAs.	0b0000
[15:12]	IESB	Indicates support for the IESB bit in the SCTLR_ELx registers. Defined values are: **0b0001** IESB bit in the SCTLR_ELx registers is supported.	0b0001

Bits	Name	Description	Reset
[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.	0b0000
[7:4]	UAO	User Access Override. Defined values are: **0b0001** UAO supported.	0b0001
[3:0]	CnP	Indicates support for Common not Private translations. Defined values are: **0b0001** Common not Private translations supported.	0b0001

Access

MRS <Xt>, ID_AA64MMFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b010

Accessibility

MRS <Xt>, ID_AA64MMFR2_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFR2_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64MMFR2_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64MMFR2_EL1;
```

A.4.19 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-97: AArch64_ccsidr_el1 bit assignments

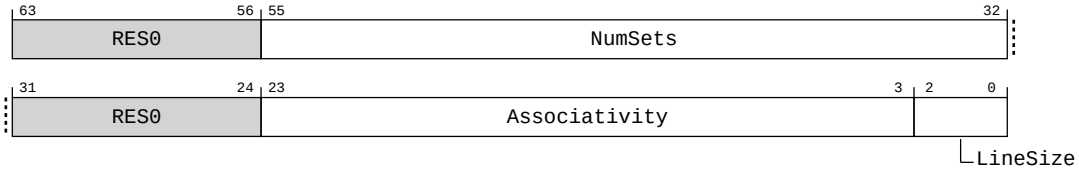


Table A-221: CCSIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[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);
    else
        return CCSIDR_EL1;
elsif PSTATE.EL == EL2 then
    return CCSIDR_EL1;
elsif PSTATE.EL == EL3 then
    return CCSIDR_EL1;
```

A.4.20 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-98: AArch64_clidr_el1 bit assignments

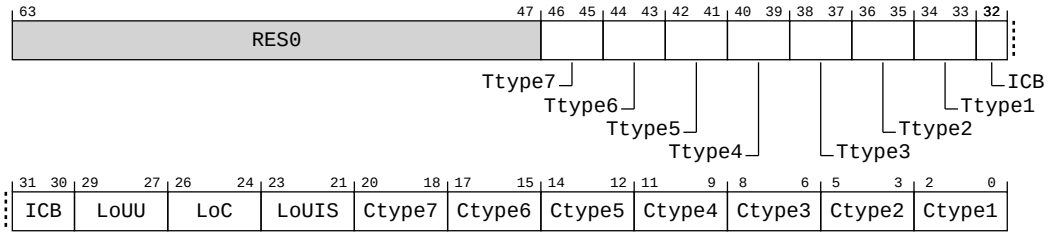


Table A-223: CLIDR_EL1 bit descriptions

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

Bits	Name	Description	Reset
[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 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 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. **0b10** Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines.	xx
[34:33]	Ttype1	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b10** Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines.	xx
[32:30]	ICB	Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions. The possible values are: **0b010** L2 cache is the highest Inner Cacheable level. This value is reported if the DSU is configured without an L3 cache. **0b011** L3 cache is the highest Inner Cacheable level. This value is reported if 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 data cache.	xxx

Bits	Name	Description	Reset
[26:24]	LoC	Level of Coherence for the cache hierarchy. **0b010** Level of Coherency is after the L2 cache. This value is reported if the DSU is configured without an L3 cache. **0b011** Level of Coherency is after the L3 cache. This value is reported if 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 data cache.	xxx
[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 the DSU is configured without an L3 cache. **0b100** Unified cache. This value is reported if 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: **0b100** Unified cache.	xxx

Bits	Name	Description	Reset
[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);
    else
        return CLIDR_EL1;
elsif PSTATE.EL == EL2 then
    return CLIDR_EL1;
elsif PSTATE.EL == EL3 then
    return CLIDR_EL1;
```

A.4.21 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 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-99: AArch64_gmid_el1 bit assignments

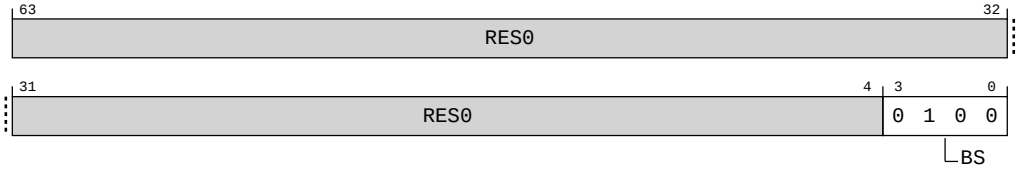


Table A-225: GMID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:4]	**RES0**	Reserved	**RES0**
[3:0]	BS	Log_2 of the block size in words. The minimum supported size is 16B (value == 2) and the maximum is 256B (value == 6). **0b0100** 64 bytes.	0b0100

Access

MRS <Xt>, GMID_EL1

op0	op1	CRn	CRm	op2
0b11	0b001	0b0000	0b0000	0b100

Accessibility

MRS <Xt>, GMID_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID5 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return GMID_EL1;
elsif PSTATE.EL == EL2 then
    return GMID_EL1;
elsif PSTATE.EL == EL3 then
    return GMID_EL1;
```

A.4.22 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-100: AArch64_csselr_el1 bit assignments

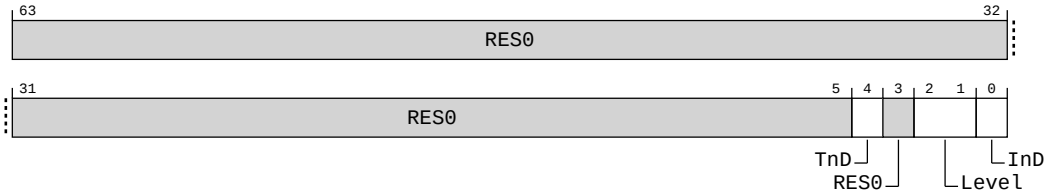


Table A-227: CSSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:5]	RES0	Reserved	RES0
[4]	TnD	Allocation Tag not Data bit. **0b0** Data, Instruction or Unified cache.	x
[3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2:1]	Level	Cache level of required cache. **0b00** Level 1 cache. **0b01** Level 2 cache. **0b10** Level 3 cache. All other values are reserved. If CSSELR_EL1.Level is programmed to a cache level that is not implemented, then the value for this field on a read of CSSELR_EL1 is **UNKNOWN**.	xx
[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);
    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);
    else
        CSSELR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    CSSELR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    CSSELR_EL1 = X[t];
```

A.4.23 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-101: AArch64_ctr_el0 bit assignments

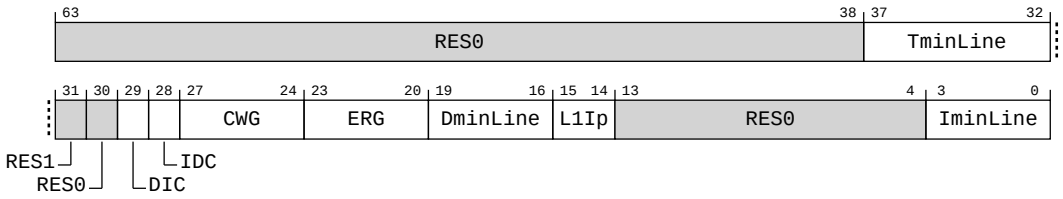


Table A-230: CTR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:38]	**RES0**	Reserved	**RES0**
[37:32]	TminLine	Tag minimum Line. Log_2 of the number of words covered by Allocation Tags in the smallest cache line of all caches which can contain Allocation tags that are controlled by the PE. **0b000100** 64 bytes.	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. Log_2 of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified. **0b0100** 64 bytes.	xxxx
[23:20]	ERG	Exclusives reservation granule. Log_2 of the number of words of the maximum size of the reservation granule for the Load-Exclusive and Store-Exclusive instructions. **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**

Bits	Name	Description	Reset
[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' && 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);
    else
        return CTR_EL0;
elsif PSTATE.EL == EL2 then
    return CTR_EL0;
elsif PSTATE.EL == EL3 then
    return CTR_EL0;
```

A.4.24 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 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-102: AArch64_dczid_el0 bit assignments

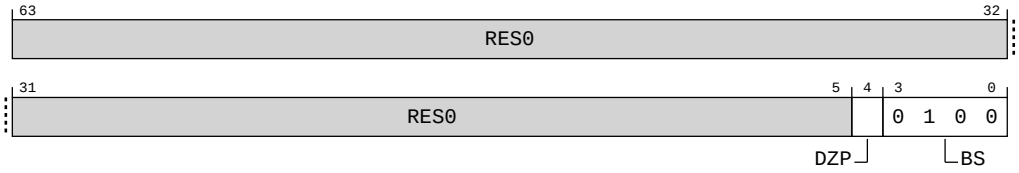


Table A-232: DCZID_EL0 bit descriptions

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.	The reset values can be the following: `0b0`, `0b1`, respective to the value.
[3:0]	BS	Log_2 of the block size in words. The maximum size supported is 2KB (value == 9). **0b0100** 64 bytes.	`0b0100`

Access

MRS <Xt>, DCZID_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b0000	0b0000	0b111

Accessibility

MRS <Xt>, DCZID_EL0

```
if PSTATE.EL == EL0 then
    return DCZID_EL0;
elsif PSTATE.EL == EL1 then
    return DCZID_EL0;
elsif PSTATE.EL == EL2 then
    return DCZID_EL0;
elsif PSTATE.EL == EL3 then
    return DCZID_EL0;
```

A.4.25 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 0000 0001 xxxx xxxx xxx0 011x 0000 0000 0011 1111


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

MPAMIDR_EL1 indicates the MPAM implementation parameters of the PE.

Figure A-103: AArch64_mpamidr_el1 bit assignments

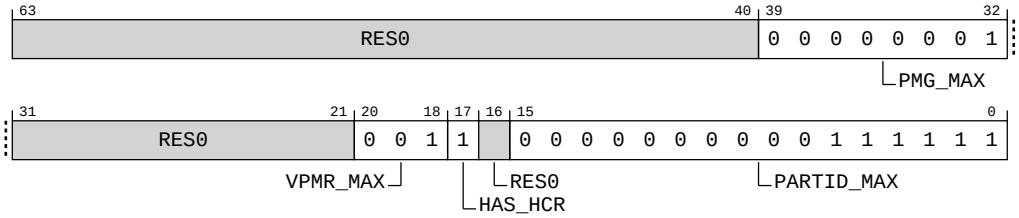


Table A-234: MPAMIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:40]	**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	0x01
[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	0b001
[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.	0b1
[16]	**RES0**	Reserved	**RES0**
[15:0]	PARTID_MAX	The largest value of PARTID that the implementation can generate. The PARTID_I and PARTID_D fields of every MPAMn_ELx must implement at least enough bits to represent PARTID_MAX. **0b0000000000111111** Max PARTID field is 63	0x003F

Access

MRS <Xt>, MPAMIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b100

Accessibility

MRS <Xt>, MPAMIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && MPAMHCR_EL2.TRAP_MPAMIDR_EL1 == '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.4.26 IMP_CPUPPMPDPCR_EL1, Global PPMPDP Configuration Register

This register controls the aggressiveness of PDP features.

Configurations

AArch64 register IMP_CPUPPMPDPCR_EL1 bits [63:0] are architecturally mapped to External System register B.1.3 CPUPPMPDPCR, Global PPMPDP Configuration Register on page 472.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-104: AArch64_imp_cpuppmpdpcr_el1 bit assignments

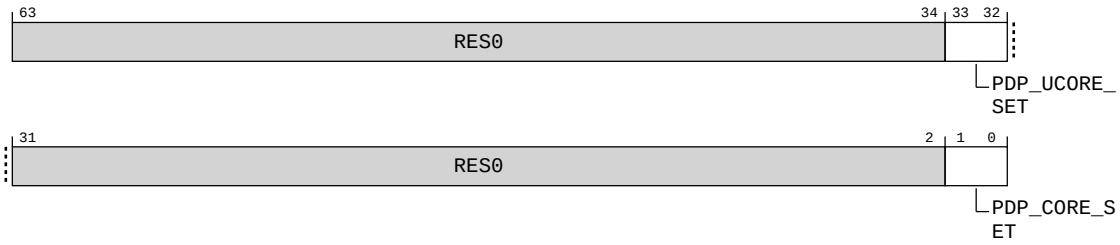


Table A-236: IMP_CPUPPMPDPCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:34]	**RES0**	Reserved	**RES0**
[33:32]	PDP_UCORE_SET	Uncore PDP Aggressiveness **0b00** Disable PDP. **0b01** Engage PDP at low aggressiveness **0b10** Engage PDP at medium aggressiveness **0b11** Engage PDP at high aggressiveness	0b00
[31:2]	**RES0**	Reserved	**RES0**
[1:0]	PDP_CORE_SET	Core PDP Aggressiveness **0b00** Disable PDP. **0b01** Engage PDP at low aggressiveness. **0b10** Engage PDP at medium aggressiveness. **0b11** Engage PDP at high aggressiveness.	0b00

Access

MRS <Xt>, S3_0_C15_C2_4

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b100

MSR S3_0_C15_C2_4, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b100

A.4.27 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 470.

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-105: AArch64_imp_cpumpmmcr_el3 bit assignments

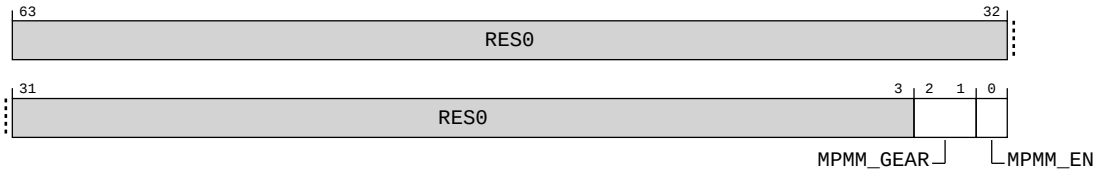


Table A-239: IMP_CPUMPMMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2:1]	MPMM_GEAR	MPMM Gear Select **0b00** Select MPMM Gear 0. **0b01** Select MPMM Gear 1. **0b10** Select MPMM Gear 2. **0b11** Select MPMM Gear 3.	0b00
[0]	MPMM_EN	MPMM Master Enable **0b0** MPMM is not enabled. **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

A.5 AArch64 Performance Monitors registers summary

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table A-242: Performance Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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
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

A.5.1 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 0000 0000 0000 0000 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-106: AArch64_pmmir_el1 bit assignments

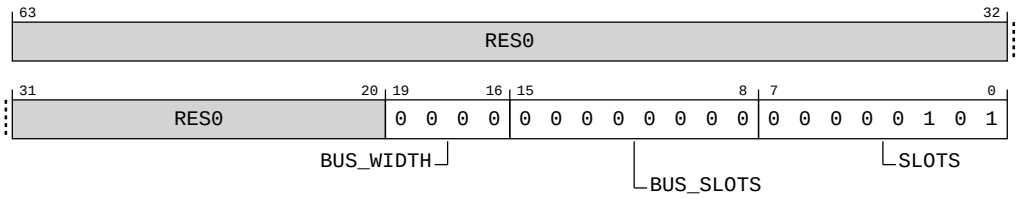


Table A-243: PMMIR_EL1 bit descriptions

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: **0b0000** The information is not available.	0b0000
[15:8]	BUS_SLOTS	Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle.	0x00

Bits	Name	Description	Reset
[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 be **RAZ**. **0b00000101** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 5.	0x05

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() && 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.5.2 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.2.28 PMCR_EL0, External Performance Monitors Control Register on page 503.

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-107: AArch64_pmcr_el0 bit assignments

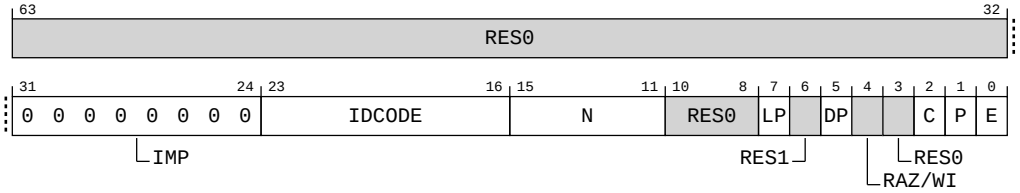


Table A-245: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	IMP	Implementer code. **0b00000000** No ID information is present in PMCR/PMCR_EL0. Software must use the MIDR_EL1 to identify the PE.	0x00
[23:16]	IDCODE	Identification code. Use of this field is deprecated. This field has an **IMPLEMENTATION DEFINED** value. Each implementer must maintain a list of identification codes that are specific to the implementer. A specific implementation is identified by the combination of the implementer code and the identification code. Access to this field is: RO	8{x}

Bits	Name	Description	Reset
[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` 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. **0b00110** Six PMU Counters Implemented Access to this field is: RO	`5{x}`
[10:8]	**RES0**	Reserved	**RES0**
[7]	LP	Long event counter enable. Determines when unsigned overflow is recorded by an event counter overflow bit. **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 EL2 is implemented and AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN is less than PMCR_EL0.N, this bit does not affect the operation of event counters in the range [AArch32-HDCR.HPMN..(PMCR_EL0.N-1)] or [AArch64-MDCR_EL2.HPMN..(PMCR_EL0.N-1)]. **Note:** The effect of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state. For more information, see the description of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN.	`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 bit. **0b1** When event counting for counters in the range [0..(AArch64-MDCR_EL2.HPMN-1)] is prohibited, cycle counting by AArch64-PMCCNTR_EL0 is disabled. 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. The effects of writing to this bit are: **0b0** No action. **0b1** Reset all event counters accessible in the current Exception level, not including AArch64-PMCCNTR_EL0, to zero. In EL0 and EL1: • If EL2 is implemented and enabled in the current Security state, and AArch64-MDCR_EL2.HPMN is less than PMCR_EL0.N, a write of 1 to this bit does not reset event counters in the range [AArch64-MDCR_EL2.HPMN..(PMCR_EL0.N-1)]. • If EL2 is not implemented, EL2 is disabled in the current Security state, or AArch64-MDCR_EL2.HPMN equals PMCR_EL0.N, 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. **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. **0b0** All event counters in the range [0..(PMN-1)] and AArch64-PMCCNTR_EL0, are disabled. **0b1** All event counters in the range [0..(PMN-1)] and AArch64-PMCCNTR_EL0, are enabled by AArch64-PMCNTENSET_EL0. If EL2 is implemented, then: • If EL2 is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. • If PMN is less than PMCR_EL0.N, this bit does not affect the operation of event counters in the range [PMN..(PMCR_EL0.N-1)]. If EL2 is not implemented, PMN is PMCR_EL0.N. **Note:** The effect of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state. For more information, see the description of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN.	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

A.5.3 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.2.29 PMCEID0, Performance Monitors Common Event Identification register 0 on page 507.

AArch64 register PMCEID0_EL0 bits [31:0] are architecturally mapped to External System register B.2.31 PMCEID2, Performance Monitors Common Event Identification register 2 on page 515.

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-108: AArch64_pmceid0_el0 bit assignments

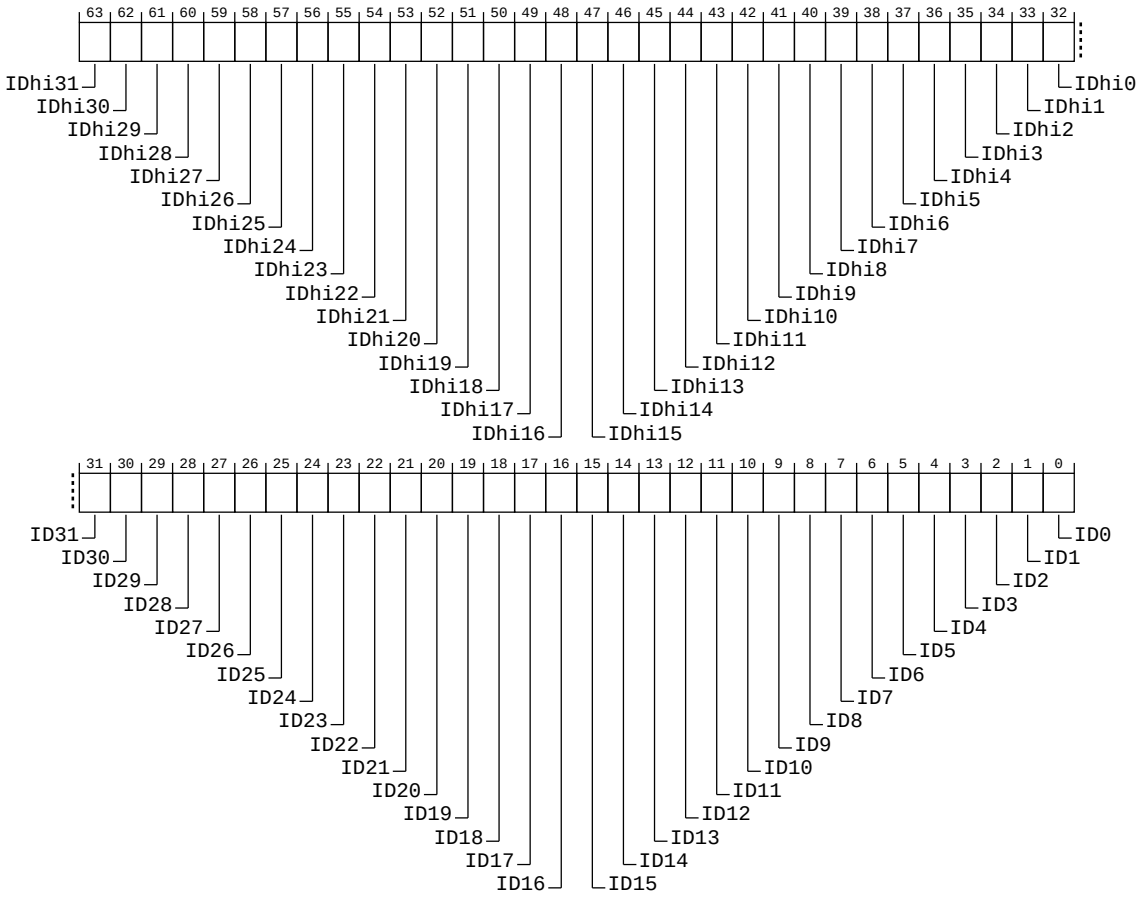


Table A-248: PMCEID0_EL0 bit descriptions

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 **0b0** The common event is not implemented, or not counted.	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 **0b1** The common event is implemented.	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 **0b1** The common event is implemented.	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 **0b1** The common event is implemented.	x
[35]	IDhi3	IDhi3 corresponds to common event (`0x4003`) SAMPLE_COLLISION **0b0** The common event is not implemented, or not counted. This value is reported if the SPE is not implemented. **0b1** The common event is implemented. This value is reported if SPE is implemented.	x
[34]	IDhi2	IDhi2 corresponds to common event (`0x4002`) SAMPLE_FILTRATE **0b0** The common event is not implemented, or not counted. This value is reported if the SPE is not implemented. **0b1** The common event is implemented. This value is reported if SPE is implemented.	x

Bits	Name	Description	Reset
[33]	IDhi1	IDhi1 corresponds to common event (0x4001) SAMPLE_FEED **0b0** The common event is not implemented, or not counted. This value is reported if the SPE is not implemented. **0b1** The common event is implemented. This value is reported if SPE is implemented.	x
[32]	IDhi0	IDhi0 corresponds to common event (0x4000) SAMPLE_POP **0b0** The common event is not implemented, or not counted. This value is reported if the SPE is not implemented. **0b1** The common event is implemented. This value is reported if SPE is implemented.	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 **0b0** The common event is not implemented, or not counted.	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 **0b1** The common event is implemented.	x
[23]	ID23	ID23 corresponds to common event (0x17) L2D_CACHE_REFILL **0b1** The common event is implemented.	x
[22]	ID22	ID22 corresponds to common event (0x16) L2D_CACHE **0b1** The common event is implemented.	x

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[7]	ID7	ID7 corresponds to common event (0x7) ST_RETIRED **0b0** The common event is not implemented, or not counted.	x
[6]	ID6	ID6 corresponds to common event (0x6) LD_RETIRED **0b0** The common event is not implemented, or not counted.	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

A.5.4 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.2.30 PMCEID1, Performance Monitors Common Event Identification register 1 on page 511.

AArch64 register PMCEID1_EL0 bits [31:0] are architecturally mapped to External System register B.2.32 PMCEID3, Performance Monitors Common Event Identification register 3 on page 519.

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-109: AArch64_pmceid1_el0 bit assignments

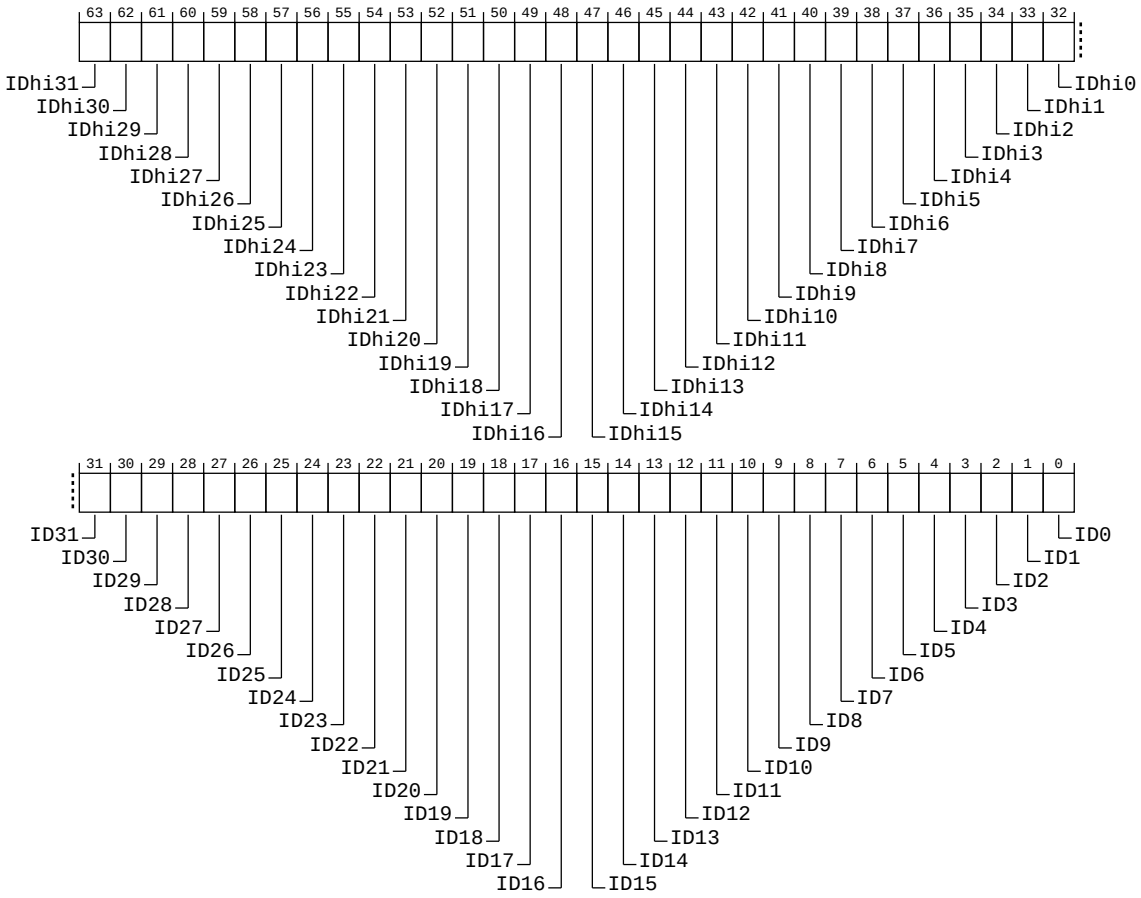


Table A-250: PMCEID1_EL0 bit descriptions

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_CHECKED_WR **0b1** The common event is implemented.	x
[37]	IDhi5	IDhi5 corresponds to common event (0x4025) MEM_ACCESS_CHECKED_RD **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 **0b0** The common event is not implemented, or not counted.	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 **0b1** The common event is implemented.	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 **0b1** The common event is implemented.	x
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL **0b1** The common event is implemented.	x
[9]	ID9	ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE **0b1** The common event is implemented.	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

Bits	Name	Description	Reset
[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 **0b1** The common event is implemented.	x

Access

MRS <Xt>, PMCEID1_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b111

A.6 AArch64 GIC system registers summary

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table A-252: GIC system registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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_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
ICH_VTR_EL2	3	4	C12	C11	1	–	64-bit	Interrupt Controller VGIC Type Register
ICC_CTLR_EL3	3	6	C12	C12	4	–	64-bit	Interrupt Controller Control Register (EL3)

A.6.1 ICC_CTLR_EL1, Interrupt Controller Control Register (EL1)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC 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-110: AArch64_icc_ctlr_el1 bit assignments

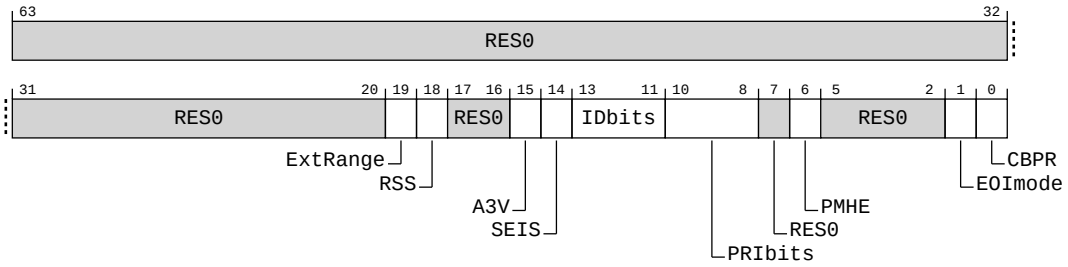


Table A-253: ICC_CTLR_EL1 bit descriptions

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.6.2 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register

Controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC 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-111: AArch64_icv_ctlr_el1 bit assignments

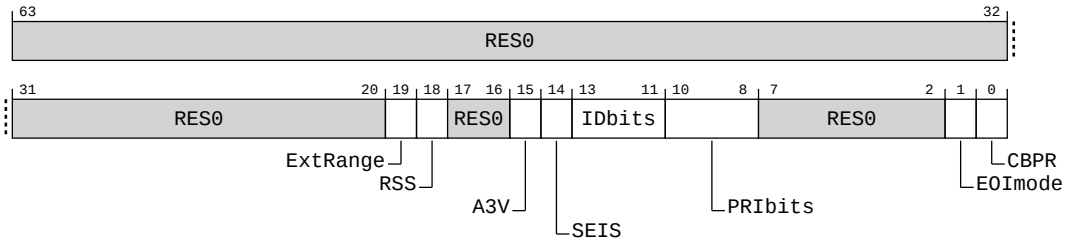


Table A-256: ICV_CTLR_EL1 bit descriptions

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.6.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Registers

Provides information about Group 0 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC 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-112: AArch64_icc_ap0r0_el1 bit assignments

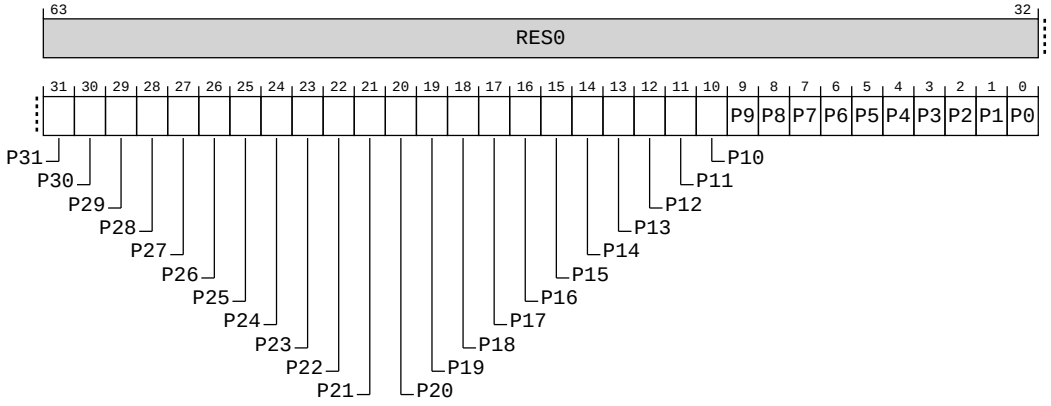


Table A-259: ICC_AP0R0_EL1 bit descriptions

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

A.6.4 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers

Provides information about virtual Group 0 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC 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-113: AArch64_icv_ap0r0_el1 bit assignments

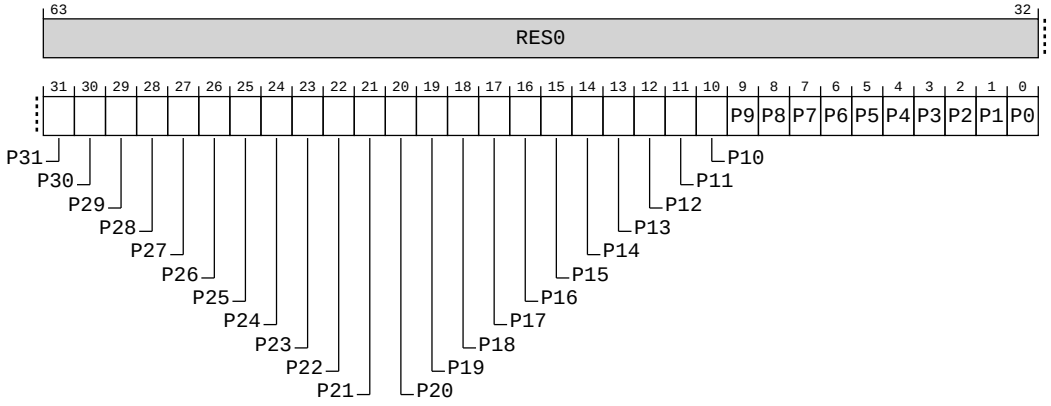


Table A-262: ICV_AP0R0_EL1 bit descriptions

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

A.6.5 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Registers

Provides information about Group 1 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC 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-114: AArch64_icc_ap1r0_el1 bit assignments

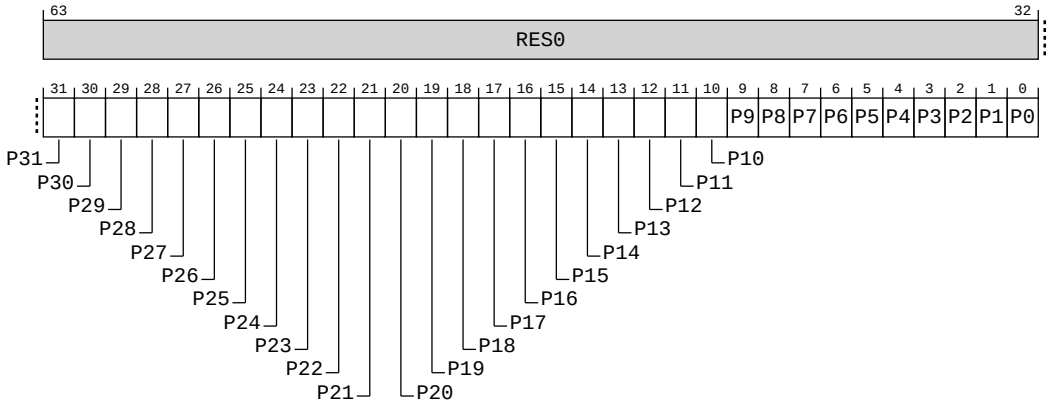


Table A-265: ICC_AP1R0_EL1 bit descriptions

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

A.6.6 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers

Provides information about virtual Group 1 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC 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-115: AArch64_icv_ap1r0_el1 bit assignments

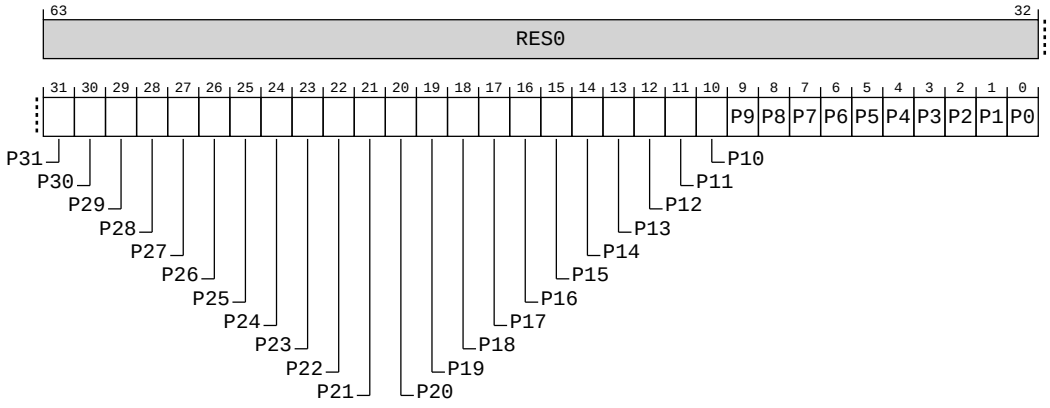


Table A-268: ICV_AP1R0_EL1 bit descriptions

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

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-116: AArch64_ich_vtr_el2 bit assignments

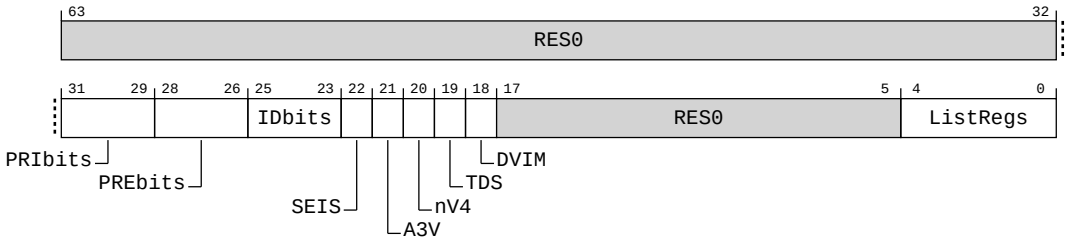


Table A-271: ICH_VTR_EL2 bit descriptions

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
[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 preemption bits are implemented	xxx
[25:23]	IDbits	The number of virtual interrupt identifier bits supported: **0b000** 16 bits.	xxx

Bits	Name	Description	Reset
[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. **0b1** Masking of Directly-injected Virtual Interrupts is 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-117: AArch64_icc_ctlr_el3 bit assignments

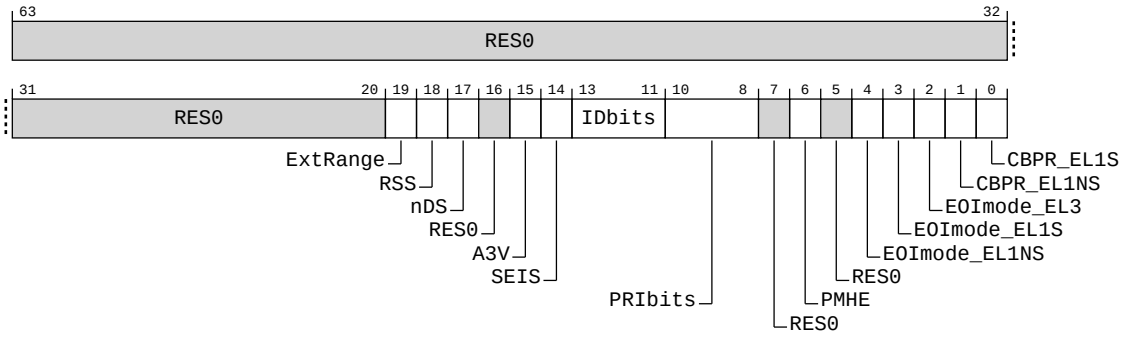


Table A-273: ICC_CTLR_EL3 bit descriptions

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. **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
[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**

Bits	Name	Description	Reset
[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`
[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`

Bits	Name	Description	Reset
[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 Other system control registers summary

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table A-276: Other system control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRBIDR_EL1	3	0	C9	C11	7	—	64-bit	Trace Buffer ID Register

A.7.1 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

Other system control registers

Access type

See bit descriptions

Reset value

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-118: AArch64_trbidr_el1 bit assignments

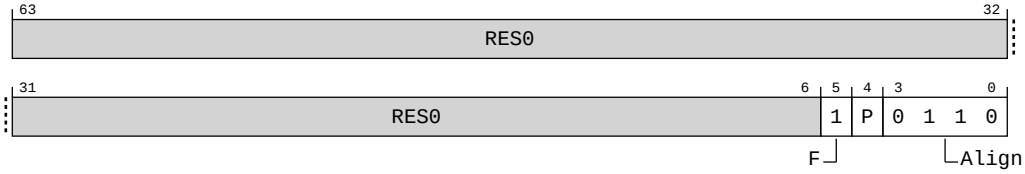


Table A-277: TRBIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:6]	**RES0**	Reserved	**RES0**
[5]	F	Flag Updates. Defines whether the address translation performed by the Trace Buffer Unit manages the Access Flag and dirty state. Defined values are: **0b1** Trace buffer address translation manages the Access Flag and dirty state in the same way as the MMU on this PE.	0b1

Bits	Name	Description	Reset
[4]	P	Programming not allowed. The trace buffer is owned by a higher Exception level or by the other Security state. Defined values are: **0b0** The owning Exception level is the current Exception level or a lower Exception level, and the owning Security state is the current Security state. **0b1** The owning Exception level is a higher Exception level, or the owning Security state is not the current Security state. The value read from this field depends on the current Exception level and the values of AArch64-MDCR_EL3.NSTB and AArch64-MDCR_EL2.E2TB: • If EL3 is implemented and either AArch64-MDCR_EL3.NSTB == 0b00 or AArch64-MDCR_EL3.NSTB == 0b01, meaning the owning Security state is Secure state, this field reads as one from: ◦ Non-secure EL2. ◦ Non-secure EL1. ◦ If Secure EL2 is implemented and enabled, and AArch64-MDCR_EL2.E2TB == 0b00, Secure EL1. • If EL3 is implemented and either AArch64-MDCR_EL3.NSTB == 0b10 or AArch64-MDCR_EL3.NSTB == 0b11, meaning 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 == 0b00, Non-secure EL1. • Otherwise, this field reads as zero.	The reset values can be the following: 0b0, 0b1, respective to the value.
[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
    return TRBIDR_EL1;
elsif PSTATE.EL == EL2 then
    return TRBIDR_EL1;
elsif PSTATE.EL == EL3 then
    return TRBIDR_EL1;
```

A.8 AArch64 RAS registers summary

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table A-279: 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

A.8.1 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 0000 0000 0000 0010

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-119: AArch64_erridr_el1 bit assignments

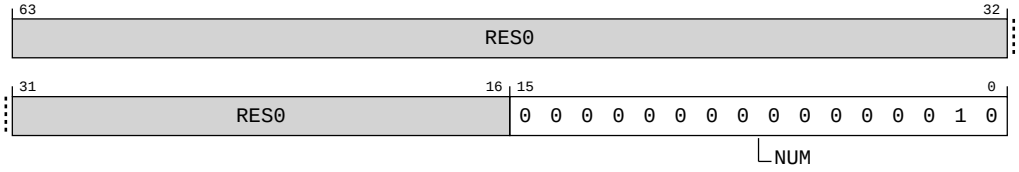


Table A-280: ERRIDR_EL1 bit descriptions

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. **0b0000000000000010** Two Records Present.	0x0002

Access

MRS <Xt>, ERRIDR_EL1

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

Accessibility

MRS <Xt>, ERRIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '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.8.2 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 xxx0

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-120: AArch64_errselr_el1 bit assignments

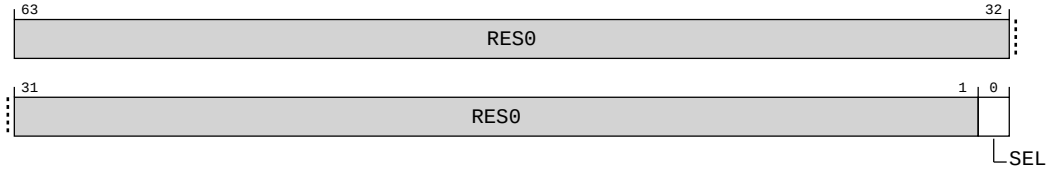


Table A-282: ERRSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	**RES0**	Reserved	**RES0**
[0]	SEL	**0b0** Selects record 0, containing errors from DSU RAMs **0b1** Selects record 1, containing errors from Core RAMs	0b0

Access

MRS <Xt>, ERRSELR_EL1

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

MSR ERRSELR_EL1, <Xt>

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

Accessibility

MRS <Xt>, ERRSELR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '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 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.8.3 ERXFR_EL1, Selected Error Record Feature Register

Accesses ext-ERR<n>FR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 0001 0000 1010 1001 1010 0010


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-121: AArch64_erxfr_el1 bit assignments

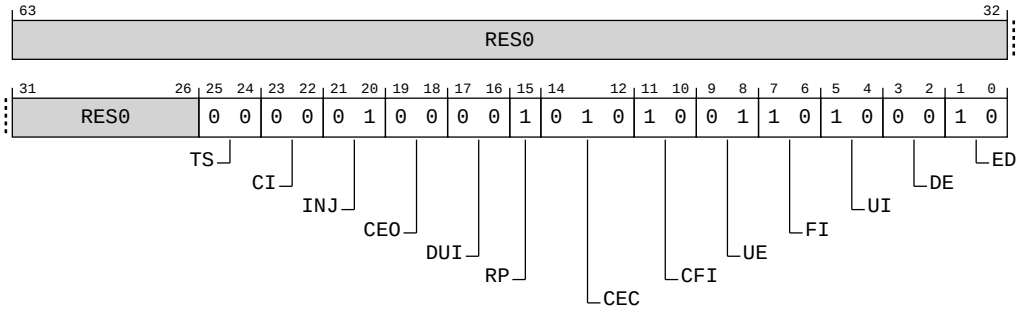


Table A-285: ERXFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:26]	**RES0**	Reserved	**RES0**
[25:24]	TS	Timestamp Extension. Indicates whether, for each error record <m> owned by this node, AArch64-ERXMISC3_EL1 is used as the timestamp register, and, if it is, the timebase used by the timestamp. **0b00** Does not support a timestamp register. All other values are reserved.	0b00
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented by the node. **0b00** Does not support the critical error interrupt. AArch64-ERXCTLR_EL1.CI is **RES0**. All other values are reserved.	0b00
[21:20]	INJ	Fault Injection Extension. Indicates whether the Common Fault Injection Model Extension is implemented by the node. **0b01** Supports the Common Fault Injection Model Extension. See AArch64-ERXPFGF_EL1 for more information. All other values are reserved.	0b01

Bits	Name	Description	Reset
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior of the node when a second or subsequent Corrected error is recorded and a first Corrected error has previously been recorded by an error record <m> owned by the node. **0b00** Keeps the previous error syndrome. All other values are reserved. The second or subsequent Corrected error is counted by the Corrected error counter, regardless of the value of this field. If counting the error causes unsigned overflow of the counter, then AArch64-ERXSTATUS_EL1.OF is set to 1. This means that, if no other error is subsequently recorded that overwrites the syndrome: • If AArch64-ERXFR_EL1.CEO is 0b00, the error record holds the syndrome for the first recorded Corrected error. • If AArch64-ERXFR_EL1.CEO is 0b01, the error record holds the syndrome for the most recently recorded Corrected error before the counter overflows.	0b00
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the enabling and disabling of error recovery interrupts on deferred errors is supported by the node. **0b00** Does not support the enabling and disabling of error recovery interrupts on deferred errors. AArch64-ERXCTLR_EL1.DUI is **RES0**. All other values are reserved.	0b00
[15]	RP	Repeat counter. Indicates whether the node implements a second Corrected error counter in AArch64-ERXMISC0_EL1 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 AArch64-ERXMISC0_EL1 for each error record <m> owned by the node that can record countable errors. The repeat counter is the same size as the primary error counter.	0b1
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter mechanisms in AArch64-ERXMISC0_EL1 for each error record <m> owned by the node that can record countable errors. **0b010** Implements an 8-bit Corrected error counter in AArch64-ERXMISC0_EL1[39:32] for each error record <m> owned by the node that can record countable errors. All other values are reserved. **Note:** Implementations might include other error counter models, or might include the standard model and not indicate this in AArch64-ERXFR_EL1.	0b010
[11:10]	CFI	Fault handling interrupt for corrected errors control. Indicates whether the enabling and disabling of fault handling interrupts on corrected errors is supported by the node. **0b10** Enabling and disabling of fault handling interrupts on corrected errors is supported and controllable using AArch64-ERXCTLR_EL1.CFI. All other values are reserved.	0b10

Bits	Name	Description	Reset
[9:8]	UE	In-band error reponse (External Abort). Indicates whether the in-band error response and associated controls are implemented by the node. **0b01** In-band error response is supported and always enabled. AArch64-ERXCTLR_EL1.UE is **RES0**. It is **IMPLEMENTATION DEFINED** whether an uncorrected error that is deferred and recorded as Deferred error, but is not deferred to the Requester, will signal an in-band error response to the Requester.	0b01
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented by the node. **0b10** Fault handling interrupt is supported and controllable using AArch64-ERXCTLR_EL1.FI.	0b10
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented by the node. **0b10** Error handling interrupt is supported and controllable using AArch64-ERXCTLR_EL1.UI.	0b10
[3:2]	DE	Defered Errors (DE). **0b00** Defered errors are always disabled	0b00
[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 AArch64-ERXCTLR_EL1.ED. All other values are reserved.	0b10

Access

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

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

MRS <Xt>, ERXFR_EL1

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

Accessibility

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXFR_EL1 is RAZ.

- Direct reads of ERXFR_EL1 are NOPs.
- Direct reads of ERXFR_EL1 are UNDEFINED.

MRS <Xt>, ERXFR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '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 ERXFR_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 ERXFR_EL1;
elsif PSTATE.EL == EL3 then
    return ERXFR_EL1;
```

A.8.4 ERXCTLR_EL1, Selected Error Record Control Register

Accesses ext-ERR<n>CTLR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS 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-122: AArch64_erxctlr_el1 bit assignments

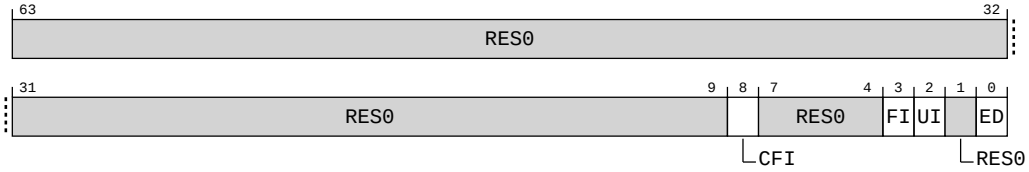


Table A-287: ERXCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:9]	**RES0**	Reserved	**RES0**
[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**

Bits	Name	Description	Reset
[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

Access

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

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

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>CTLR is not present, meaning reads and writes of ERXCTLR_EL1 are **RES0**.

MRS <Xt>, ERXCTLR_EL1

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

MSR ERXCTLR_EL1, <Xt>

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

Accessibility

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.

- ERXCTLR_EL1 is RAZ/WI.
- Direct reads and writes of ERXCTLR_EL1 are NOPs.
- Direct reads and writes of ERXCTLR_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>CTLR is not present, meaning reads and writes of ERXCTLR_EL1 are RES0.
MRS <Xt>, ERXCTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '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 ERXCTLR_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 ERXCTLR_EL1;
elsif PSTATE.EL == EL3 then
    return ERXCTLR_EL1;
```

MSR ERXCTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '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
        ERXCTLR_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
        ERXCTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
```

```
    ERXCTLR_EL1 = X[t];
```

A.8.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register

Accesses ext-ERR<n>STATUS for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 00xx x0xx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-123: AArch64_erxstatus_el1 bit assignments

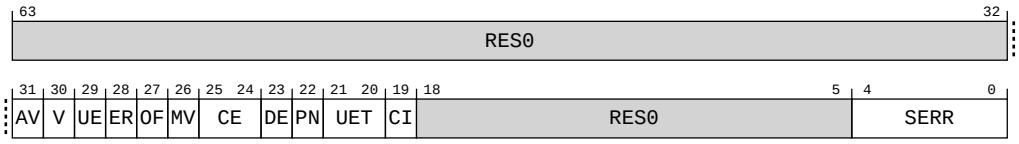


Table A-290: ERXSTATUS_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	AV	Address Valid. **0b0** ext-ERR<n>ADDR not valid. **0b1** ext-ERR<n>ADDR contains an address associated with the highest priority error recorded by this record. Access to this field is: W1C	0b0
[30]	V	Status Register Valid. **0b0** 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-ERR<n>STATUS.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-ERR<n>STATUS.UE == '0' && ext-ERR<n>STATUS.DE == '0' && this field can be set to 0b1 by a Deferred error** Access to this field is: **UNKNOWN**/WI **When ext-ERR<n>STATUS.UE == '0' && this field is never set to 0b1 by a Deferred error** Access to this field is: **UNKNOWN**/WI **When ext-ERR<n>STATUS.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-ERR<n>STATUS.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. **0b01** At least one transient error was corrected. **0b10** At least one error was corrected. **0b11** At least one persistent error was corrected. The mechanism by which a component or node detects whether a Corrected error is transient or persistent is **IMPLEMENTATION DEFINED**. If no such mechanism is implemented, then the node sets this field to `0b10` when a corrected error is recorded. 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-ERR<n>STATUS.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-ERR<n>STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x

Bits	Name	Description	Reset
[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-ERR<n>STATUS.V == '0' || (ext-ERR<n>STATUS.DE == '0' && ext-ERR<n>STATUS.UE == '0')** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x
[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-ERR<n>STATUS.V == '0' || ext-ERR<n>STATUS.UE == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	xx
[19]	CI	Critical Error. Indicates whether a critical error condition has been recorded. **0b0** No critical error condition. **0b1** Critical error condition. When clearing 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-ERR<n>STATUS.V == '0'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: W1C	x

Bits	Name	Description	Reset
[18:5]	**RES0**	Reserved	**RES0**
[4:0]	SERR	Architecturally-defined primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry. **0b00000** No error. **0b00010** Data value from (non-associative) internal memory. For example, ECC from on-chip SRAM or buffer. **0b00110** Data value from associative memory. For example, ECC error on cache data. **0b00111** Address/control value from associative memory. For example, ECC error on cache tag. **0b01000** Data value from a TLB. For example, ECC error on TLB data. **0b01001** Address/control value from a TLB. For example, ECC error on TLB tag. **0b10010** Error response from Completer of access. For example, error response from cache write-back. **0b11000** Deferred error from Requester passed through. For example, poisoned data received from the Requester of an access and deferred to the Completer. All other values are reserved. This field is not valid and reads **UNKNOWN** if ERR<n>STATUS.V == 0b0.	5{x}

Access

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXSTATUS_EL1 is RAZ/WI.
- Direct reads and writes of ERXSTATUS_EL1 are NOPs.
- Direct reads and writes of ERXSTATUS_EL1 are UNDEFINED.

ext-ERR<n>STATUS describes additional constraints that also apply when ext-ERR<n>STATUS is accessed through ERXSTATUS_EL1.

MRS <Xt>, ERXSTATUS_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b010

MSR ERXSTATUS_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b010

Accessibility

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXSTATUS_EL1 is RAZ/WI.
- Direct reads and writes of ERXSTATUS_EL1 are NOPs.
- Direct reads and writes of ERXSTATUS_EL1 are UNDEFINED.

ext-ERR<n>STATUS describes additional constraints that also apply when ext-ERR<n>STATUS is accessed through ERXSTATUS_EL1.
MRS <Xt>, ERXSTATUS_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 SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXSTATUS_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 ERXSTATUS_EL1;
elsif PSTATE.EL == EL3 then
    return ERXSTATUS_EL1;
```

MSR ERXSTATUS_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 SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        ERXSTATUS_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
        ERXSTATUS_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXSTATUS_EL1 = X[t];
```

A.8.6 ERXADDR_EL1, Selected Error Record Address Register

Accesses ext-ERR<n>ADDR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS 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-124: AArch64_erxaddr_el1 bit assignments

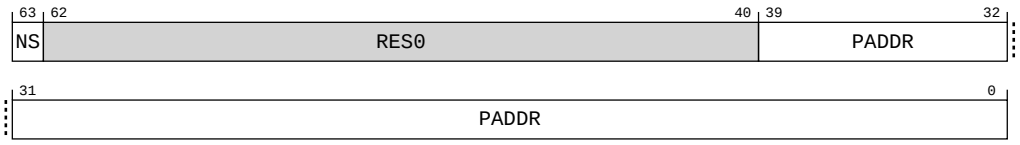


Table A-293: ERXADDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure attribute. **0b0** The address is Secure. **0b1** The address is Non-secure.	x
[62:40]	**RES0**	Reserved	**RES0**
[39:0]	PADDR	Physical Address. Address of the recorded location.	40{x}

Access

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXADDR_EL1 is RAZ/WI.
- Direct reads and writes of ERXADDR_EL1 are NOPs.
- Direct reads and writes of ERXADDR_EL1 are UNDEFINED.

ext-ERR<n>ADDR describes additional constraints that also apply when ext-ERR<n>ADDR is accessed through ERXADDR_EL1.

MRS <Xt>, ERXADDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b011

MSR ERXADDR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b011

Accessibility

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXADDR_EL1 is RAZ/WI.
- Direct reads and writes of ERXADDR_EL1 are NOPs.
- Direct reads and writes of ERXADDR_EL1 are UNDEFINED.

ext-ERR<n>ADDR describes additional constraints that also apply when ext-ERR<n>ADDR is accessed through ERXADDR_EL1.

MRS <Xt>, ERXADDR_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 SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXADDR_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 ERXADDR_EL1;
elsif PSTATE.EL == EL3 then
    return ERXADDR_EL1;
```

MSR ERXADDR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '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
        ERXADDR_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
        ERXADDR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXADDR_EL1 = X[t];
```

A.8.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register

Accesses ext-ERR<n>PFGF for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x100 xxxx xxxx xxxx xxx0 0000 0110 0010

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-125: AArch64_erxpfgf_el1 bit assignments

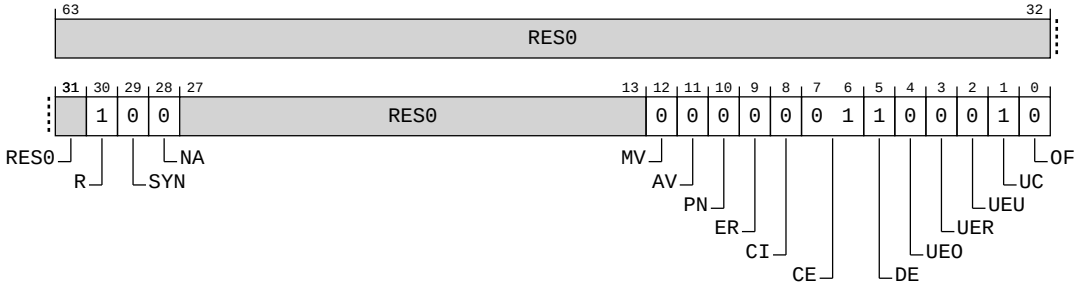


Table A-296: ERXPFGF_EL1 bit descriptions

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 AARCH64-ERXPFGCTL_EL1.R. AARCH64-ERXPFGCTL_EL1.R is a read/write field.	0b1

Bits	Name	Description	Reset
[29]	SYN	Syndrome. Fault syndrome injection. **0b0** When an injected error is recorded, the node sets AARCH64-ERXSTATUS_EL1.{IERR, SERR} to IMPLEMENTATION DEFINED values. AARCH64-ERXSTATUS_EL1.{IERR, SERR} are **UNKNOWN** when AARCH64-ERXSTATUS_EL1.V is 0. **Note:** If ERR<n>PFGF.SYN is 1, software can write specific values into the AARCH64-ERXSTATUS_EL1.{IERR, SERR} fields when setting up a fault injection event. The sets of values that can be written to these fields is **IMPLEMENTATION DEFINED**.	0b0
[28]	NA	No access required. Defines whether this component fakes detection of the error on an access to the component or spontaneously in the fault injection state. **0b0** The component fakes detection of the error on an access to the component.	0b0
[27:13]	**RES0**	Reserved	**RES0**
[12]	MV	Miscellaneous syndrome. Additional syndrome injection. Defines whether software can control all or part of the syndrome recorded in the ERR<n>MISC<m> registers when an injected error is recorded. **0b0** When an injected error is recorded, the node might update ERR<n>MISC<m>. If any syndrome is recorded by the node in ERR<n>MISC<m>, then AARCH64-ERXSTATUS_EL1.MV is set to 1. AARCH64-ERXPFGCTL_EL1.MV is **RES1**. **Note:** If ERR<n>PFGF.MV is 1, software can write specific additional syndrome values into the ERR<n>MISC<m> registers when setting up a fault injection event. The values that can be written to these registers are **IMPLEMENTATION DEFINED**.	0b0
[11]	AV	Address syndrome. Address syndrome injection. **0b0** When an injected error is recorded, the node either sets AARCH64-ERXADDR_EL1 and AARCH64-ERXSTATUS_EL1.AV for the access, or leaves these unchanged. AARCH64-ERXPFGCTL_EL1.AV is **RES0**. **Note:** If ERR<n>PFGF.AV is 1, software can write a specific address value into AARCH64-ERXADDR_EL1 when setting up a fault injection event.	0b0
[10]	PN	Poison flag. Describes how the fault generation feature of the node sets the AARCH64-ERXSTATUS_EL1.PN status flag. **0b0** When an injected error is recorded, it is IMPLEMENTATION DEFINED whether the node sets AARCH64-ERXSTATUS_EL1.PN to 1. AARCH64-ERXPFGCTL_EL1.PN is **RES0**. This behavior replaces the architecture-defined rules for setting the AARCH64-ERXSTATUS_EL1.PN bit.	0b0
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the AARCH64-ERXSTATUS_EL1.ER status flag. **0b0** When an injected error is recorded, the node sets AARCH64-ERXSTATUS_EL1.ER according to the architecture-defined rules for setting the ER field. AARCH64-ERXPFGCTL_EL1.ER is **RES0**.	0b0

Bits	Name	Description	Reset
[8]	CI	Critical Error flag. Describes how the fault generation feature of the node sets the AARCH64-ERXSTATUS_EL1.CI status flag. **0b0** When an injected error is recorded, it is IMPLEMENTATION DEFINED whether the node sets AARCH64-ERXSTATUS_EL1.CI to 1. AARCH64-ERXPFGCTL_EL1.CI is **RES0**. This behavior replaces the architecture-defined rules for setting the AARCH64-ERXSTATUS_EL1.CI bit.	0b0
[7:6]	CE	Corrected Error generation. Describes the types of Corrected error that the fault generation feature of the node can generate. **0b01** The fault generation feature of the node allows generation of a non-specific Corrected error, that is, a Corrected error that is recorded by setting AARCH64-ERXSTATUS_EL1.CE to 0b10. AARCH64-ERXPFGCTL_EL1.CE is a read/write field. The values 0b10 and 0b11 in AARCH64-ERXPFGCTL_EL1.CE are reserved. All other values are reserved. If AARCH64-ERXFR_EL1.FRX is 1, then AARCH64-ERXFR_EL1.CE indicates whether the node supports this type of error.	0b01
[5]	DE	Deferred Error generation. Describes whether the fault generation feature of the node can generate Deferred errors. **0b1** The fault generation feature of the node allows generation of Deferred errors. AARCH64-ERXPFGCTL_EL1.DE is a read/write field. If AARCH64-ERXFR_EL1.FRX is 1, then AARCH64-ERXFR_EL1.DE indicates whether the node supports this type of error.	0b1
[4]	UEO	Latent or Restartable Error generation. Describes whether the fault generation feature of the node can generate Latent or Restartable errors. **0b0** The fault generation feature of the node does not generate Latent or Restartable errors. AARCH64-ERXPFGCTL_EL1.UEO is **RES0**. If AARCH64-ERXFR_EL1.FRX is 1, then AARCH64-ERXFR_EL1.UEO indicates whether the node supports this type of error.	0b0
[3]	UER	Signaled or Recoverable Error generation. Describes whether the fault generation feature of the node can generate Signaled or Recoverable errors. **0b0** The fault generation feature of the node does not generate Signaled or Recoverable errors. AARCH64-ERXPFGCTL_EL1.UER is **RES0**. If AARCH64-ERXFR_EL1.FRX is 1, then AARCH64-ERXFR_EL1.UER indicates whether the node supports this type of error.	0b0

Bits	Name	Description	Reset
[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. AARCH64-ERXPFGCTL_EL1.UEU is **RES0**. If AARCH64-ERXFR.FRX_EL1 is 1, then AARCH64-ERXFR_EL1.UEU indicates whether the node supports this type of error.	0b0
[1]	UC	Uncontainable Error generation. Describes whether the fault generation feature of the node can generate Uncontainable errors. **0b1** The fault generation feature of the node allows generation of Uncontainable errors. AARCH64-ERXPFGCTL_EL1.UC is a read/write field. If AARCH64-ERXFR.FRX_EL1 is 1, then AARCH64-ERXFR_EL1.UC indicates whether the node supports this type of error.	0b1
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the AARCH64-ERXSTATUS_EL1.OF status flag. **0b0** When an injected error is recorded, the node sets AARCH64-ERXSTATUS_EL1.OF according to the architecture-defined rules for setting the OF field. AARCH64-ERXPFGCTL_EL1.OF is **RES0**.	0b0

Access

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXPFGF_EL1 is RAZ.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the RAS Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGF_EL1 is RAZ.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are **UNDEFINED**.


Note

A node does not implement the RAS Common Fault Injection Model Extension when ERR<q>FR.INJ == 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGF is not present, meaning reads of ERXPFGF_EL1 are **RES0**.

ext-ERR<n>PFGF describes additional constraints that also apply when ext-ERR<n>PFGF is accessed through ERXPFGF_EL1.

MRS <Xt>, ERXPFGF_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b100

Accessibility

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXPFGF_EL1 is RAZ.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the RAS Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGF_EL1 is RAZ.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are UNDEFINED.


Note

A node does not implement the RAS Common Fault Injection Model Extension when ERR<q>FR.INJ == 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGF is not present, meaning reads of ERXPFGF_EL1 are RES0.

ext-ERR<n>PFGF describes additional constraints that also apply when ext-ERR<n>PFGF is accessed through ERXPFGF_EL1.
MRS <Xt>, ERXPFGF_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGF_EL1;
```

```
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGF_EL1;
elsif PSTATE.EL == EL3 then
    return ERXPFGF_EL1;
```

A.8.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register

Accesses ext-ERR<n>PFGCTL for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS 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-126: AArch64_erxpfgctl_el1 bit assignments

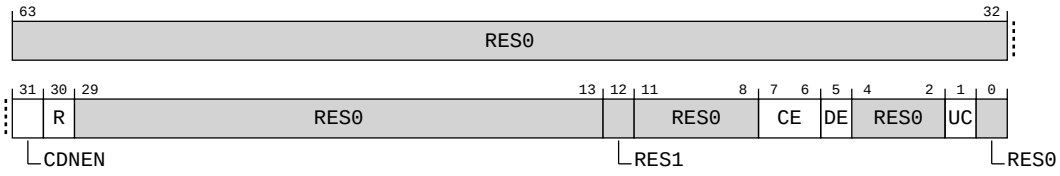


Table A-298: ERXPFGCTL_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	CDNEN	Countdown Enable. Controls transfers of the value held in AArch64-ERXPFGCDN_EL1 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 AArch64-ERXPFGCDN_EL1.CDN.	x
[30]	R	Restartable. Support for Error Generation Counter restart mode. **0b0** The node does not support this feature. **0b1** Feature controllable.	x
[29:13]	**RES0**	Reserved	**RES0**
[12]	**RES1**	Reserved	**RES1**
[11:8]	**RES0**	Reserved	**RES0**
[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 cannot generate this type of error. **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 AArch64-ERXSTATUS_EL1.CE == 0b10. All other values are reserved. If AArch64-ERXFR_EL1.FRX is 0b1 then AArch64-ERXFR_EL1.CE indicates whether the node supports this type of error. This field reads-as-zeros if the node does not support this type of error.	xx
[5]	DE	Deferred Error generation. Describes whether the fault generation feature of the node can generate this type of error. **0b0** The fault generation feature of the node cannot generate this type of error. **0b1** The fault generation feature of the node allows generation of this type of error. If AArch64-ERXFR_EL1.FRX is 0b1 then AArch64-ERXFR_EL1.DE indicates whether the node supports this type of error. This bit reads-as-zero if the node does not support this type of error.	x
[4:2]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[1]	UC	Uncontainable Error generation. Describes whether the fault generation feature of the node can generate this type of error. **0b0** The fault generation feature of the node cannot generate this type of error. **0b1** The fault generation feature of the node allows generation of this type of error. If AArch64-ERXFR_EL1.FRX is 0b1 then AArch64-ERXFR_EL1.UC indicates whether the node supports this type of error. This bit reads-as-zero if the node does not support this type of error.	x
[0]	**RES0**	Reserved	**RES0**

Access

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXPFGCTL_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the RAS Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCTL_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are **UNDEFINED**.


Note

A node does not implement the RAS Common Fault Injection Model Extension when ERR<q>FR.INJ == 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGCTL is not present, meaning reads and writes of ERXPFGCTL_EL1 are **RES0**.

ext-ERR<n>PFGCTL describes additional constraints that also apply when ext-ERR<n>PFGCTL is accessed through ERXPFGCTL_EL1.

MRS <Xt>, ERXPFGCTL_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b101

MSR ERXPFGCTL_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b101

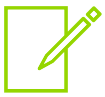
Accessibility

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXPFGCTL_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the RAS Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCTL_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are UNDEFINED.


Note

A node does not implement the RAS Common Fault Injection Model Extension when ERR<q>FR.INJ == 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGCTL is not present, meaning reads and writes of ERXPFGCTL_EL1 are RES0.

ext-ERR<n>PFGCTL describes additional constraints that also apply when ext-ERR<n>PFGCTL is accessed through ERXPFGCTL_EL1.
MRS <Xt>, ERXPFGCTL_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGCTL_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif SCR_EL3.FIEN == '0' then
```

```
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGCTL_EL1;
elsif PSTATE.EL == EL3 then
    return ERXPFGCTL_EL1;
```

MSR ERXPFGCTL_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCTL_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCTL_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXPFGCTL_EL1 = X[t];
```

A.8.9 ERXPFGCDN_EL1, Selected Pseudo-fault Generation Countdown register

Accesses ext-ERR<n>PFGCDN for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS 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-127: AArch64_erxpfgcdn_el1 bit assignments

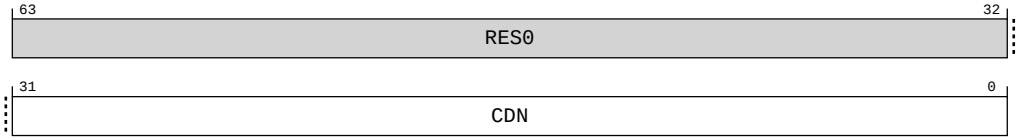


Table A-301: ERXPFGCDN_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	CDN	Countdown value. This field is copied to Error Generation Counter when either: • Software writes ext-ERR<n>PFGCTL.CDNEN with 1. • The Error Generation Counter decrements to zero and ext-ERR<n>PFGCTL.R == 1. While ext-ERR<n>PFGCTL.CDNEN == 1 and the Error Generation Counter is nonzero, the counter decrements by 1 for each cycle at an **IMPLEMENTATION DEFINED** clock rate. When the counter reaches 0, one of the errors enabled in the ext-ERR<n>PFGCTL register is generated. **Note:** The current Error Generation Counter value is not visible to software.	32{x}

Access

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXPFGCDN_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the RAS Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCDN_EL1 is RAZ/WI.

- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are **UNDEFINED**.


Note

A node does not implement the RAS Common Fault Injection Model Extension when ERR<q>FR.INJ == 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGCDN is not present, meaning reads and writes of ERXPFGCDN_EL1 are **RES0**.

ext-ERR<n>PFGCDN describes additional constraints that also apply when ext-ERR<n>PFGCDN is accessed through ERXPFGCDN_EL1.

MRS <Xt>, ERXPFGCDN_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b110

MSR ERXPFGCDN_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b110

Accessibility

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXPFGCDN_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the RAS Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCDN_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are UNDEFINED.


Note

A node does not implement the RAS Common Fault Injection Model Extension when ERR<q>FR.INJ == 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGCDN is not present, meaning reads and writes of ERXPFGCDN_EL1 are RES0.

ext-ERR<n>PFGCDN describes additional constraints that also apply when ext-ERR<n>PFGCDN is accessed through ERXPFGCDN_EL1.
MRS <Xt>, ERXPFGCDN_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGCDN_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGCDN_EL1;
elsif PSTATE.EL == EL3 then
    return ERXPFGCDN_EL1;
```

MSR ERXPFGCDN_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXPFGCDN_EL1 = X[t];
```

A.8.10 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0

Accesses ext-ERR<n>MISC0 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-128: AArch64_erxmisc0_el1 bit assignments

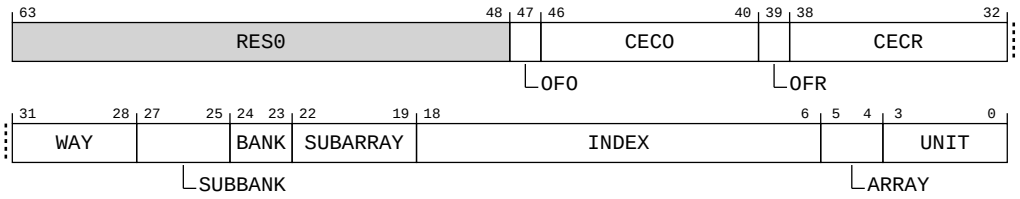


Table A-304: ERXMISC0_EL1 bit descriptions

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

Bits	Name	Description	Reset
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERXMISC0_EL1.CECO is incremented and wraps through zero. **0b0** Other counter has not overflowed. **0b1** Other counter has overflowed. A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Warm reset.	0b0
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERXMISC0_EL1.CECR. Unaffected by Warm reset.	0b0000000
[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERXMISC0_EL1.CECR is incremented and wraps through zero. **0b0** Repeat counter has not overflowed. **0b1** Repeat counter has overflowed. A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Warm reset.	0b0
[38:32]	CECR	Corrected error count, repeat. Incremented for the first countable error, which also records other syndrome for the error, and subsequently for each countable error that matches the recorded other syndrome.	0b0000000

Bits	Name	Description	Reset
[31:28]	WAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L1 Data Cache] • Indicates which Tag RAM way or data RAM way detected the error. Upper 2 bits are unused. [L2 TLB] • Indicates which RAM detected an error. The possible values are 0 (RAM 1) to 9 (RAM 10). [L1 Instruction Cache] • Indicates which way detected the error. [L2 Tag Cache] • Indicates which way detected the error. Upper 1 bit unused. [L2 Data Cache] • Unused. Unaffected by Warm reset.	0b0000
[27:25]	SUBBANK	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L1 Instruction Cache] • Indicates which subbank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero. Unaffected by Warm reset.	0b000
[24:23]	BANK	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which L2 bank detected the error. Upper 1 bit is unused. [L1 Instruction Cache] • Indicates which bank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero. Upper 1 bit is unused Unaffected by Warm reset.	0b00

Bits	Name	Description	Reset
[22:19]	SUBARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which L2 data doubleword detected the error. Upper 1 bit is unused. [L1 Data Cache] • Indicates for L1 Data RAM which word had the error detected. For L1 Tag RAMs which bank had the error (0b0000: bank0 , 0b0001: bank1) [L2 TLB] • Indicates for L2 TLB RAM which word had the error detected. Upper 3 bits are unsued. Unaffected by Warm reset.	0b0000
[18:6]	INDEX	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Tag Cache] • Indicates which index detected the error. Bits[n:6] are the index, n varies with the cache size. [L2 Data Cache] • Indicates which index detected the error. Bits[n+3:3] are the index, and bits[2:0] are the way, n varies wit the cache size. [L1 Data Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size [L2 TLB] • Index of TLB RAM. Upper 4 bits are unused. [L1 Instruction Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. Unaffected by Warm reset.	0b0000000000000

Bits	Name	Description	Reset
[5:4]	ARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] Indicates which array detected the error. The possible values are: • `0b00` L2 Tag RAM. • `0b01` L2 Data RAM. • `0b10` Transaction Queue RAM. [L1 Data Cache] Indicates which array detected the error. The possible values are: • 00 LS Tag RAM 0. • 01 LS Tag RAM 1. • 10 LS Data RAM. • 11 LS Tag RAM 2. [L1 Instruction Cache] Indicates which array that detected the error, Data Array has higher priority. The possible values are: • `0b01` Data. Unaffected by Warm reset.	`0b00`
[3:0]	UNIT	Indicates the unit which detected the error. The possible values are: **`0b0001`** L1 Instruction Cache. **`0b0010`** L2 TLB. **`0b0100`** L1 Data Cache. **`0b1000`** L2 Cache.	`0b0000`

Access

If AArch64-ERRIDR_EL1.NUM == `0x0000` or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC0_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC0_EL1 are NOPs.
- Direct reads and writes of ERXMISC0_EL1 are UNDEFINED.

ext-ERR<n>MISC0 describes additional constraints that also apply when ext-ERR<n>MISC0 is accessed through ERXMISC0_EL1.

MRS <Xt>, ERXMISC0_EL1

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

MSR ERXMISC0_EL1, <Xt>

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

Accessibility

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC0_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC0_EL1 are NOPs.
- Direct reads and writes of ERXMISC0_EL1 are UNDEFINED.

ext-ERR<n>MISC0 describes additional constraints that also apply when ext-ERR<n>MISC0 is accessed through ERXMISC0_EL1.
MRS <Xt>, ERXMISC0_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 SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC0_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 ERXMISC0_EL1;
elsif PSTATE.EL == EL3 then
    return ERXMISC0_EL1;
```

MSR ERXMISC0_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 SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC0_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
        ERXMISC0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXMISC0_EL1 = X[t];
```

A.8.11 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1

Accesses ext-ERR<n>MISC1 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS 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-129: AArch64_erxmisc1_el1 bit assignments

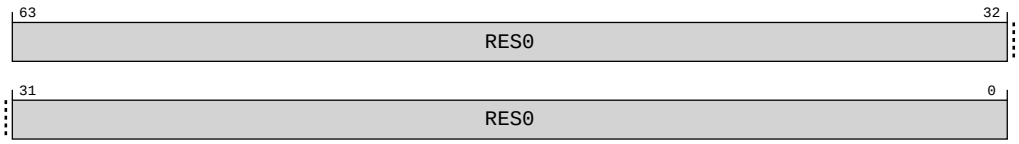


Table A-307: ERXMISC1_EL1 bit descriptions

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

Access

If AArch64-ERRIDR_EL1.NUM == `0x0000` or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC1_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC1_EL1 are NOPs.
- Direct reads and writes of ERXMISC1_EL1 are UNDEFINED.

ext-ERR<n>MISC1 describes additional constraints that also apply when ext-ERR<n>MISC1 is accessed through ERXMISC1_EL1.

MRS <Xt>, ERXMISC1_EL1

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

MSR ERXMISC1_EL1, <Xt>

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

Accessibility

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC1_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC1_EL1 are NOPs.
- Direct reads and writes of ERXMISC1_EL1 are UNDEFINED.

ext-ERR<n>MISC1 describes additional constraints that also apply when ext-ERR<n>MISC1 is accessed through ERXMISC1_EL1.
MRS <Xt>, ERXMISC1_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 SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC1_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 ERXMISC1_EL1;
elsif PSTATE.EL == EL3 then
    return ERXMISC1_EL1;
```

MSR ERXMISC1_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 SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC1_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
        ERXMISC1_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXMISC1_EL1 = X[t];
```

A.8.12 ERXMISC2_EL1, Selected Error Record Miscellaneous Register 2

Accesses ext-ERR<n>MISC2 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS 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-130: AArch64_erxmisc2_el1 bit assignments

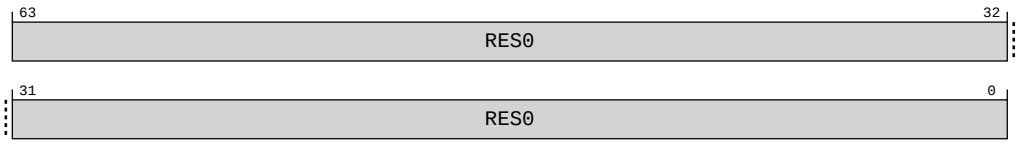


Table A-310: ERXMISC2_EL1 bit descriptions

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

Access

If AArch64-ERRIDR_EL1.NUM == `0x0000` or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC2_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC2_EL1 are NOPs.

- Direct reads and writes of ERXMISC2_EL1 are UNDEFINED.

ext-ERR<n>MISC2 describes additional constraints that also apply when ext-ERR<n>MISC2 is accessed through ERXMISC2_EL1.

MRS <Xt>, ERXMISC2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b010

MSR ERXMISC2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b010

Accessibility

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC2_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC2_EL1 are NOPs.
- Direct reads and writes of ERXMISC2_EL1 are UNDEFINED.

ext-ERR<n>MISC2 describes additional constraints that also apply when ext-ERR<n>MISC2 is accessed through ERXMISC2_EL1.
MRS <Xt>, ERXMISC2_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 SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC2_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 ERXMISC2_EL1;
elsif PSTATE.EL == EL3 then
    return ERXMISC2_EL1;
```

MSR ERXMISC2_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '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
        ERXMISC2_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
        ERXMISC2_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXMISC2_EL1 = X[t];
```

A.8.13 ERXMISC3_EL1, Selected Error Record Miscellaneous Register 3

Accesses ext-ERR<n>MISC3 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS 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-131: AArch64_erxmisc3_el1 bit assignments

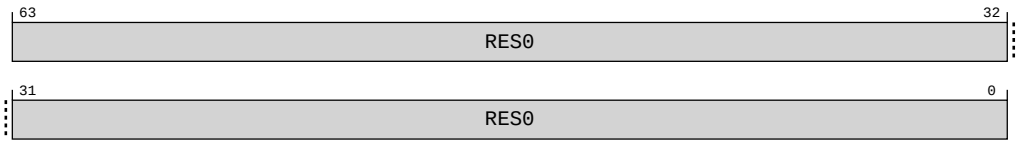


Table A-313: ERXMISC3_EL1 bit descriptions

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

Access

If AArch64-ERRIDR_EL1.NUM == `0x0000` or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC3_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC3_EL1 are NOPs.
- Direct reads and writes of ERXMISC3_EL1 are UNDEFINED.

ext-ERR<n>MISC3 describes additional constraints that also apply when ext-ERR<n>MISC3 is accessed through ERXMISC3_EL1.

MRS <Xt>, ERXMISC3_EL1

op0	op1	CRn	CRm	op2
`0b11`	`0b000`	`0b0101`	`0b0101`	`0b011`

MSR ERXMISC3_EL1, <Xt>

op0	op1	CRn	CRm	op2
`0b11`	`0b000`	`0b0101`	`0b0101`	`0b011`

Accessibility

If AArch64-ERRIDR_EL1.NUM == 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC3_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC3_EL1 are NOPs.
- Direct reads and writes of ERXMISC3_EL1 are UNDEFINED.

ext-ERR<n>MISC3 describes additional constraints that also apply when ext-ERR<n>MISC3 is accessed through ERXMISC3_EL1.
MRS <Xt>, ERXMISC3_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 SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC3_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 ERXMISC3_EL1;
elsif PSTATE.EL == EL3 then
    return ERXMISC3_EL1;
```

MSR ERXMISC3_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && EDSCR.SDD == '1' && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '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
        ERXMISC3_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
        ERXMISC3_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXMISC3_EL1 = X[t];
```

A.9 AArch64 Activity Monitors registers summary

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table A-316: Activity Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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
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
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

A.9.1 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.4.5 AMEVTYPER10, Activity Monitors Event Type Registers 1 on page 578.

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 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-132: AArch64_amevtyper10_el0 bit assignments

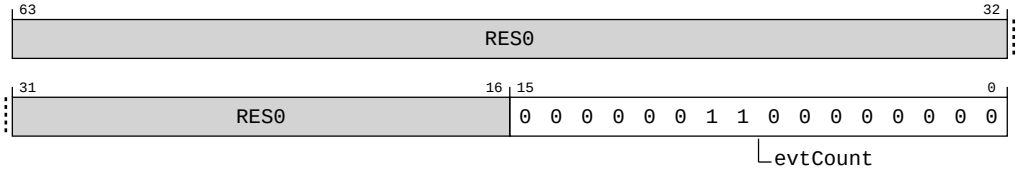


Table A-317: AMEVTYPER10_EL0 bit descriptions

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	0x0300

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

A.9.2 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.4.6 AMEVTYPER11, Activity Monitors Event Type Registers 1 on page 579.

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 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-133: AArch64_amevtyper11_el0 bit assignments

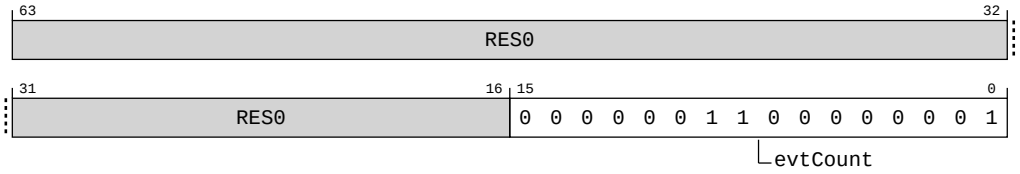


Table A-319: AMEVTYPER11_EL0 bit descriptions

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	0x0301

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

A.9.3 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.4.7 AMEVTYPER12, Activity Monitors Event Type Registers 1 on page 580.

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 0010


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-134: AArch64_amevtyper12_el0 bit assignments

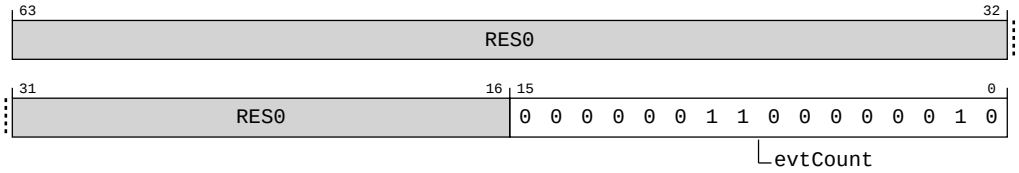


Table A-321: AMEVTYPER12_EL0 bit descriptions

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	0x0302

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>, AMEVTYPER12_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b010

A.9.4 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.4.9 AMCFGR, Activity Monitors Configuration Register on page 582.

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-135: AArch64_amcfgr_el0 bit assignments

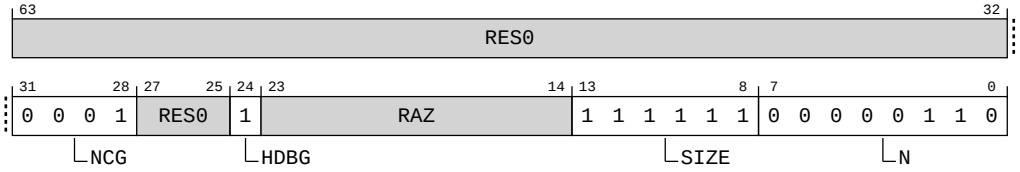


Table A-323: AMCFGR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:28]	NCG	Defines the number of counter groups. The following value is specified for this product. **0b0001** Two counter groups are implemented	0b0001
[27:25]	**RES0**	Reserved	**RES0**
[24]	HDBG	Halt-on-debug supported. From Armv8, 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**

Bits	Name	Description	Reset
[13:8]	SIZE	Defines the size of activity monitor event counters. The size of the activity monitor event counters implemented by the activity monitors Extension is defined as [AMCFGR_EL0.SIZE + 1]. From Armv8, the counters are 64-bit, and so this field is `0b111111`. **Note:** Software also uses this field to determine the spacing of counters in the memory-map. From Armv8, the counters are at doubleword-aligned addresses. **`0b111111`** 64 bits	`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 defined as [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`

A.9.5 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.4.8 AMCGCR, Activity Monitors Counter Group Configuration Register on page 581.

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-136: AArch64_amcgcr_el0 bit assignments

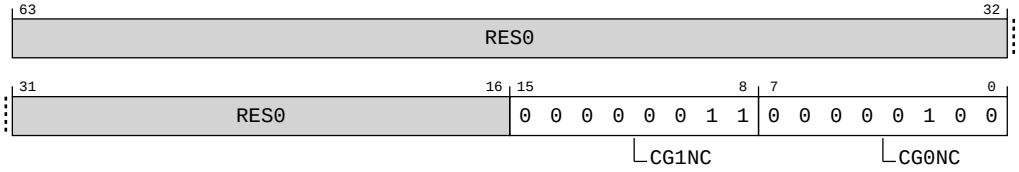


Table A-325: AMCGCR_EL0 bit descriptions

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. In an implementation that includes FEAT_AMUv1, the value of this field is `0x4`. **`0b00000100`** Four counters in the architected counter group	`0x04`

Access

MRS <Xt>, AMCGCR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0010	0b010

A.9.6 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.4.1 AMEVTYPER00, Activity Monitors Event Type Registers 0 on page 573.

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-137: AArch64_amevtyper00_el0 bit assignments

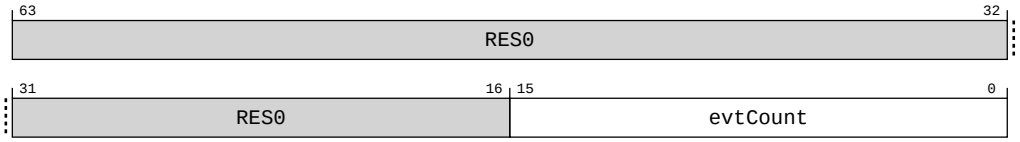


Table A-327: AMEVTYPER00_EL0 bit descriptions

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 required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0000000000010001** Processor frequency cycles	The reset values can be the following: 0b0000000000010001, respective to the value.

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

A.9.7 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.4.2 AMEVTYPER01, Activity Monitors Event Type Registers 0 on page 574.

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-138: AArch64_amevtyper01_el0 bit assignments

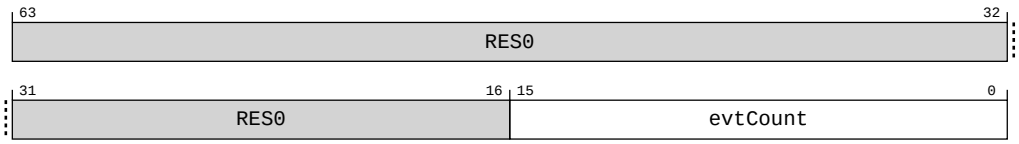


Table A-329: AMEVTYPER01_EL0 bit descriptions

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 required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0100000000000100** Constant frequency cycles	The reset values can be the following: 0b0100000000000100, respective to the value.

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

A.9.8 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.4.3 AMEVTYPER02, Activity Monitors Event Type Registers 0 on page 575.

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-139: AArch64_amevtyper02_el0 bit assignments

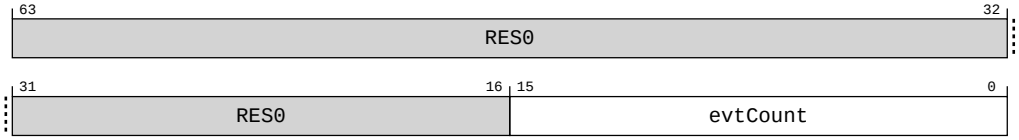


Table A-331: AMEVTYPER02_EL0 bit descriptions

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 required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0000000000001000** Instructions retired	The reset values can be the following: 0b0000000000001000, respective to the value.

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>, AMEVTYPER02_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b010

A.9.9 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.4.4 AMEVTYPER03, Activity Monitors Event Type Registers 0 on page 576.

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-140: AArch64_amevtyper03_el0 bit assignments

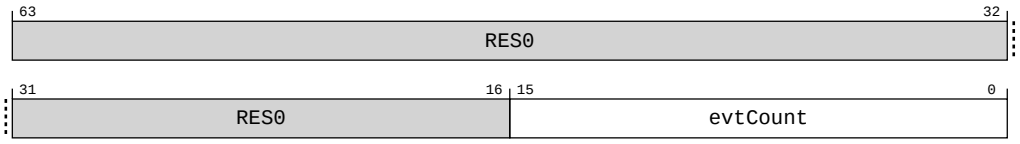


Table A-333: AMEVTYPER03_EL0 bit descriptions

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 required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0100000000000101** Memory stall cycles	The reset values can be the following: 0b0100000000000101, respective to the value.

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>, AMEVTYPER03_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b011

A.10 AArch64 Trace registers summary

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table A-335: Trace registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCIDR8	2	1	C0	C0	6	–	64-bit	ID Register 8
TRCIMSPEC0	2	1	C0	C0	7	–	64-bit	IMP DEF Register 0
TRCIDR9	2	1	C0	C1	6	–	64-bit	ID Register 9
TRCIDR2	2	1	C0	C10	7	–	64-bit	ID Register 2
TRCIDR3	2	1	C0	C11	7	–	64-bit	ID Register 3
TRCIDR4	2	1	C0	C12	7	–	64-bit	ID Register 4
TRCIDR5	2	1	C0	C13	7	–	64-bit	ID Register 5

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCIDR6	2	1	C0	C14	7	–	64-bit	ID Register 6
TRCIDR7	2	1	C0	C15	7	–	64-bit	ID Register 7
TRCIDR10	2	1	C0	C2	6	–	64-bit	ID Register 10
TRCIDR11	2	1	C0	C3	6	–	64-bit	ID Register 11
TRCIDR12	2	1	C0	C4	6	–	64-bit	ID Register 12
TRCIDR13	2	1	C0	C5	6	–	64-bit	ID Register 13
TRCAUXCTLR	2	1	C0	C6	0	–	64-bit	Auxiliary Control Register
TRCIDR0	2	1	C0	C8	7	–	64-bit	ID Register 0
TRCIDR1	2	1	C0	C9	7	–	64-bit	ID Register 1
TRCDEVARCH	2	1	C7	C15	6	–	64-bit	Device Architecture Register
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

A.10.1 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.5.2 TRCIDR8, External ID Register 8 on page 599.

Attributes

Width

64

Functional group

Trace registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-141: AArch64_trcidr8 bit assignments

Table A-336: 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**	0x00000000

Access

MRS <Xt>, TRCIDR8

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b110

A.10.2 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.5.8 TRCIMSPEC0, External IMP DEF Register 0 on page 606.

Attributes

Width

64

Functional group

Trace registers

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-142: AArch64_trcimspec0 bit assignments

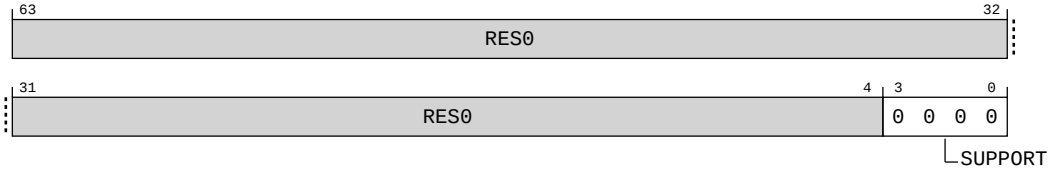


Table A-338: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[63:4]	**RES0**	Reserved	**RES0**
[3:0]	SUPPORT	Indicates whether the implementation supports **IMPLEMENTATION DEFINED** features. **0b0000** No **IMPLEMENTATION DEFINED** features are supported.	0b0000

Access

MRS <Xt>, TRCIMSPEC0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b111

MSR TRCIMSPEC0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b111

A.10.3 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.5.3 TRCIDR9, External ID Register 9 on page 601.

Attributes

Width

64

Functional group

Trace 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-143: AArch64_trcidr9 bit assignments

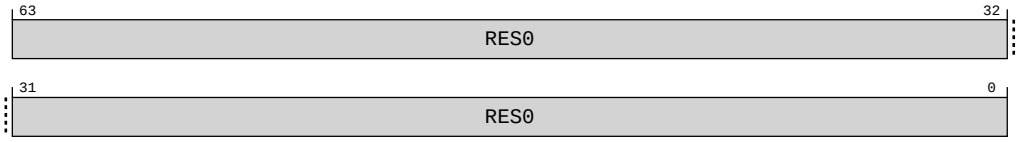


Table A-341: TRCIDR9 bit descriptions

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

Access

MRS <Xt>, TRCIDR9

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b110

A.10.4 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.5.11 TRCIDR2, External ID Register 2 on page 611.

Attributes

Width

64

Functional group

Trace registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 1100 000x xxxx xxxx x001 0000 1000 1000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-144: AArch64_trcidr2 bit assignments

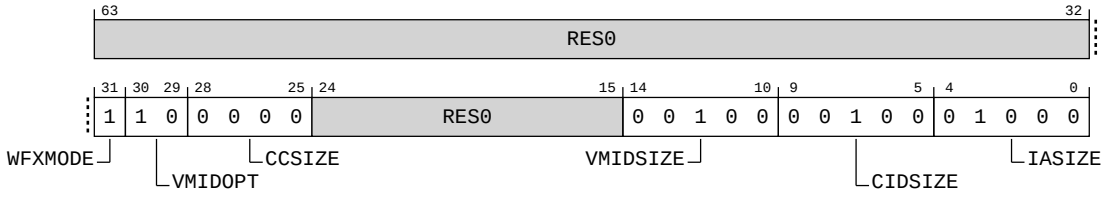


Table A-343: TRCIDR2 bit descriptions

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.	0b1
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. **0b10** Virtual context identifier selection not supported. AArch64-TRCCONFIGR.VMIDOPT is **RES1**.	0b10
[28:25]	CCSIZE	Indicates the size of the cycle counter. **0b0000** The cycle counter is 12 bits in length.	0b0000
[24:15]	**RES0**	Reserved	**RES0**
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. **0b00100** 32-bit Virtual context identifier size.	0b00100
[9:5]	CIDSIZE	Indicates the Context identifier size. **0b00100** 32-bit Context identifier size.	0b00100

Bits	Name	Description	Reset
[4:0]	IASIZE	Virtual instruction address size. **0b01000** Maximum of 64-bit instruction address size.	0b01000

Access

MRS <Xt>, TRCIDR2

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1010	0b111

A.10.5 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.5.12 TRCIDR3, External ID Register 3 on page 613.

Attributes

Width

64

Functional group

Trace registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0001 x111 1111 xx00 0000 0000 0100

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-145: AArch64_trcidr3 bit assignments

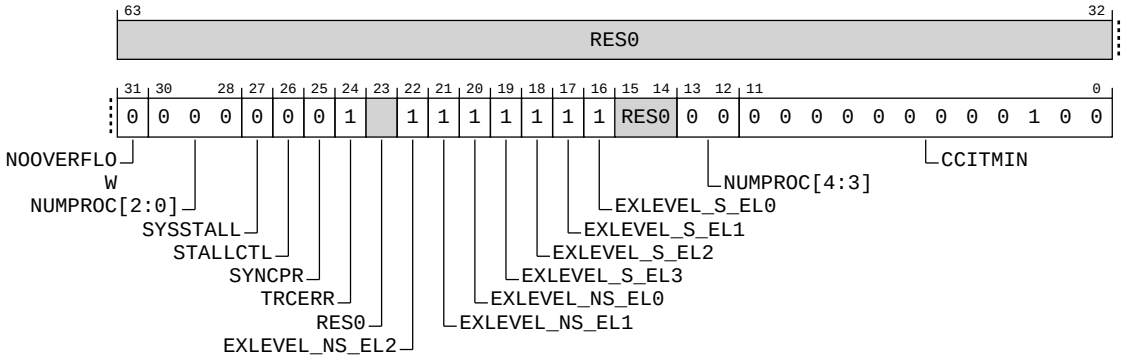


Table A-345: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	0b0
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **0b0** Stalling of the PE is not permitted.	0b0
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **0b0** Stalling of the PE is not implemented.	0b0
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0b0** AArch64-TRCSYNCPR is read/write so software can change the synchronization period.	0b0
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **0b1** Forced tracing of System Error exceptions is implemented.	0b1
[23]	**RES0**	Reserved	**RES0**
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 is implemented. **0b1** Non-secure EL2 is implemented.	0b1
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 is implemented. **0b1** Non-secure EL1 is implemented.	0b1
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 is implemented. **0b1** Non-secure EL0 is implemented.	0b1

Bits	Name	Description	Reset
[19]	EXLEVEL_S_EL3	Indicates if EL3 is implemented. **0b1** EL3 is implemented.	0b1
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 is implemented. **0b1** Secure EL2 is implemented.	0b1
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 is implemented. **0b1** Secure EL1 is implemented.	0b1
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 is implemented. **0b1** Secure EL0 is implemented.	0b1
[15:14]	**RES0**	Reserved	**RES0**
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. **0b00000** The trace unit can trace one PE.	0b00000
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in 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**	0x004

Access

MRS <Xt>, TRCIDR3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1011	0b111

A.10.6 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.5.13 TRCIDR4, External ID Register 4 on page 616.

Attributes

Width

64

Functional group

Trace registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0001 0001 0001 0111 0000 xxx0 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-146: AArch64_trcidr4 bit assignments

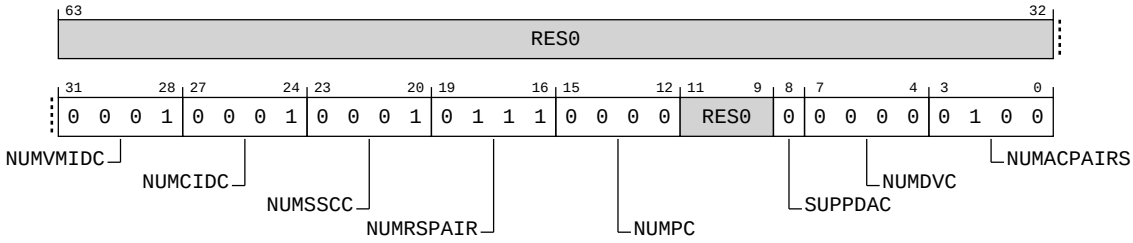


Table A-347: TRCIDR4 bit descriptions

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.	0b0001
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Context Identifier Comparator.	0b0001
[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.	0b0001
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0b0111** The implementation has eight resource selector pairs.	0b0111
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0b0000** No PE Comparator Inputs are available.	0b0000
[11:9]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[8]	SUPPDAC	Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0** Data address comparisons not implemented.	0b0
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0000** No data value comparators implemented.	0b0000
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **0b0100** The implementation has four Address Comparator pairs.	0b0100

Access

MRS <Xt>, TRCIDR4

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1100	0b111

A.10.7 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.5.14 TRCIDR5, External ID Register 5 on page 617.

Attributes

Width

64

Functional group

Trace registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x010 100x 0100 0111 xxxx 1001 1111 1111


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-147: AArch64_trcidr5 bit assignments

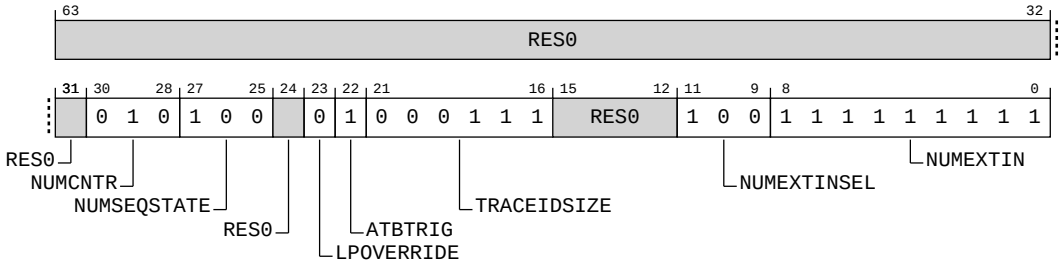


Table A-349: TRCIDR5 bit descriptions

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.	0b010
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **0b100** Four Sequencer states are implemented.	0b100
[24]	**RES0**	Reserved	**RES0**
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **0b0** The trace unit does not support Low-power Override Mode.	0b0
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **0b1** The implementation supports ATB triggers.	0b1
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **0b000111** The implementation supports a 7-bit trace ID.	0b000111
[15:12]	**RES0**	Reserved	**RES0**
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. **0b100** 4 External Input Selector resources are available.	0b100
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **0b111111111** Unified PMU event selection.	0b111111111

Access

MRS <Xt>, TRCIDR5

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1101	0b111

A.10.8 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.5.15 TRCIDR6, External ID Register 6 on page 619.

Attributes

Width

64

Functional group

Trace 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-148: AArch64_trcidr6 bit assignments

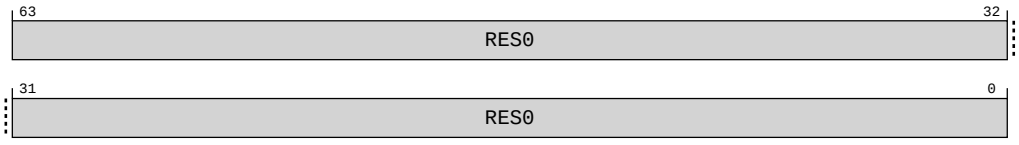


Table A-351: TRCIDR6 bit descriptions

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

Access

MRS <Xt>, TRCIDR6

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1110	0b111

A.10.9 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.5.16 TRCIDR7, External ID Register 7 on page 620.

Attributes

Width

64

Functional group

Trace 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-149: AArch64_trcidr7 bit assignments

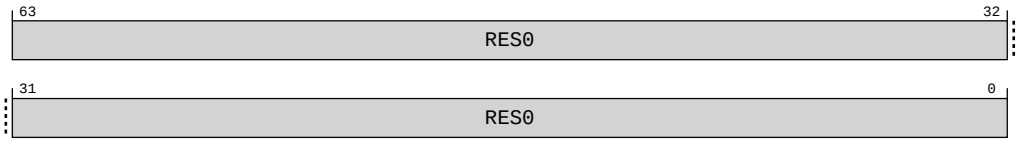


Table A-353: TRCIDR7 bit descriptions

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

Access

MRS <Xt>, TRCIDR7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1111	0b111

A.10.10 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.5.4 TRCIDR10, External ID Register 10 on page 602.

Attributes

Width

64

Functional group

Trace 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-150: AArch64_trcidr10 bit assignments

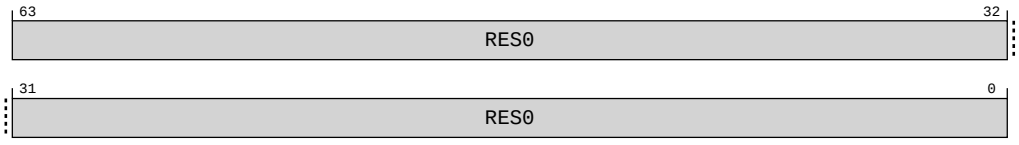


Table A-355: TRCIDR10 bit descriptions

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

Access

MRS <Xt>, TRCIDR10

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b110

A.10.11 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.5.5 TRCIDR11, External ID Register 11 on page 603.

Attributes

Width

64

Functional group

Trace 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-151: AArch64_trcidr11 bit assignments

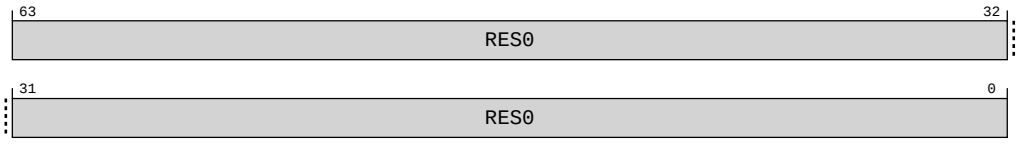


Table A-357: TRCIDR11 bit descriptions

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

Access

MRS <Xt>, TRCIDR11

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0011	0b110

A.10.12 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.5.6 TRCIDR12, External ID Register 12 on page 604.

Attributes

Width

64

Functional group

Trace 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-152: AArch64_trcidr12 bit assignments

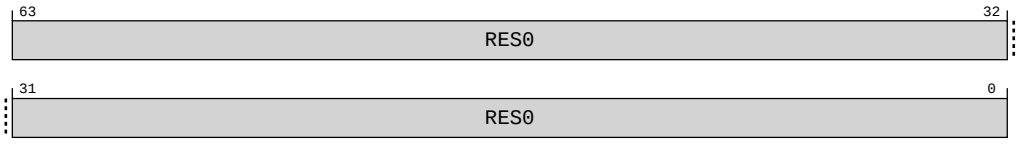


Table A-359: TRCIDR12 bit descriptions

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

Access

MRS <Xt>, TRCIDR12

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0100	0b110

A.10.13 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.5.7 TRCIDR13, External ID Register 13 on page 605.

Attributes

Width

64

Functional group

Trace 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-153: AArch64_trcidr13 bit assignments

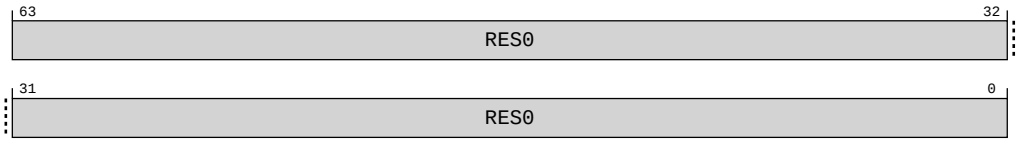


Table A-361: TRCIDR13 bit descriptions

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

Access

MRS <Xt>, TRCIDR13

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0101	0b110

A.10.14 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.5.1 TRCAUXCTLR, External Auxiliary Control Register on page 598.

Attributes

Width

64

Functional group

Trace 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-154: AArch64_trcauxctlr bit assignments

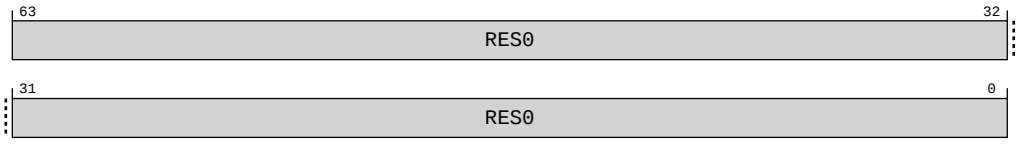


Table A-363: TRCAUXCTLR bit descriptions

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

A.10.15 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.5.9 TRCIDR0, External ID Register 0 on page 608.

Attributes

Width

64

Functional group

Trace registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x010 1000 xxxx xxx0 00xx 111x 1010 000x


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-155: AArch64_trcidr0 bit assignments

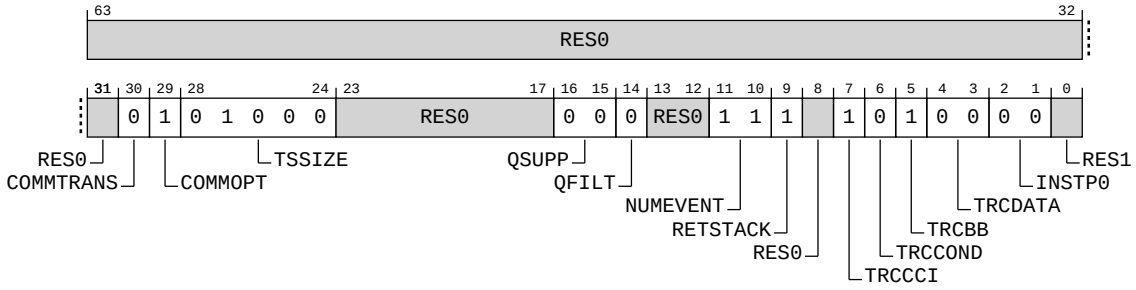


Table A-366: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30]	COMMTRANS	Transaction Start element behavior. **0b0** Transaction Start elements are P0 elements.	0b0
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **0b1** Commit mode 1.	0b1
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. **0b01000** Global timestamping implemented with a 64-bit timestamp value.	0b01000
[23:17]	**RES0**	Reserved	**RES0**
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **0b00** Q element support is not implemented.	0b00
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	0b0
[13:12]	**RES0**	Reserved	**RES0**
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **0b11** The trace unit supports 4 ETEEvents.	0b11
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **0b1** Return stack implemented.	0b1
[8]	**RES0**	Reserved	**RES0**
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **0b1** Cycle counting implemented.	0b1

Bits	Name	Description	Reset
[6]	TRCCOND	Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b0** Conditional instruction tracing not implemented.	0b0
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. **0b1** Branch broadcasting implemented.	0b1
[4:3]	TRCDATA	Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Tracing of data addresses and data values is not implemented.	0b00
[2:1]	INSTP0	Indicates if load and store instructions are P0 instructions. Load and store instructions as P0 instructions is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Load and store instructions are not P0 instructions.	0b00
[0]	**RES1**	Reserved	**RES1**

Access

MRS <Xt>, TRCIDR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b111

A.10.16 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.5.10 TRCIDR1, External ID Register 1 on page 610.

Attributes

Width

64

Functional group

Trace registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0100 0001 xxxx xxxx xxxx 1111 1111 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-156: AArch64_trcidr1 bit assignments

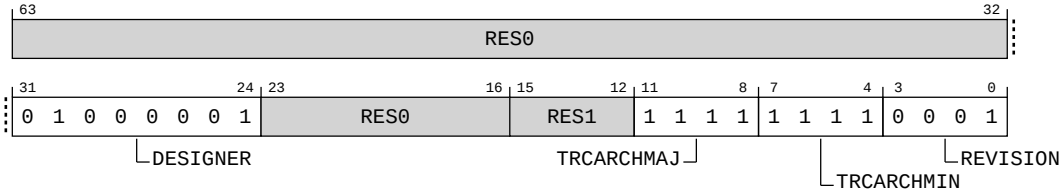


Table A-368: TRCIDR1 bit descriptions

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. **0b01000001** Arm Limited	0x41
[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.	0b1111
[7:4]	TRCARCHMIN	Minor architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH.	0b1111
[3:0]	REVISION	Indicates the major revision of the product **0b0001** r1p2	0b0001

Access

MRS <Xt>, TRCIDR1

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b111

A.10.17 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.5.20 TRCDEVARCH, External Device Architecture Register on page 627.

Attributes

Width

64

Functional group

Trace 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-157: AArch64_trcdevarch bit assignments

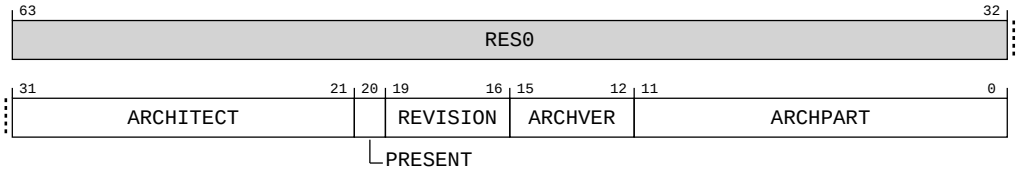


Table A-370: TRCDEVARCH bit descriptions

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. **0b0000** ETEv1.0, FEAT_ETE. All other values are reserved.	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

A.10.18 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.5.23 TRCDEVID, External Device Configuration Register on page 631.

Attributes

Width

64

Functional group

Trace 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-158: AArch64_trcdevid bit assignments

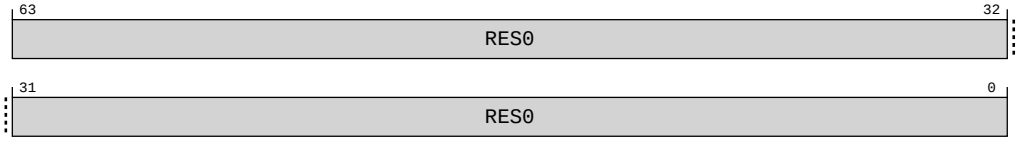


Table A-372: TRCDEVID bit descriptions

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

Access

MRS <Xt>, TRCDEVID

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b0010	0b111

A.10.19 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.5.18 TRCCLAIMSET, External Claim Tag Set Register on page 623.

Attributes

Width

64

Functional group

Trace registers

Access type

RAOW1S

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0000 0000 0000 0000 0000 1111

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-159: AArch64_trcclaimset bit assignments

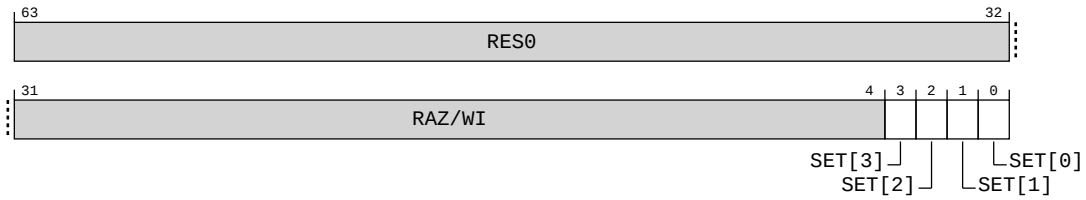


Table A-374: TRCCLAIMSET bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

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
[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

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

A.10.20 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.5.19 TRCCLAIMCLR, External Claim Tag Clear Register on page 625.

Attributes

Width

64

Functional group

Trace 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-160: AArch64_trcclaimclr bit assignments

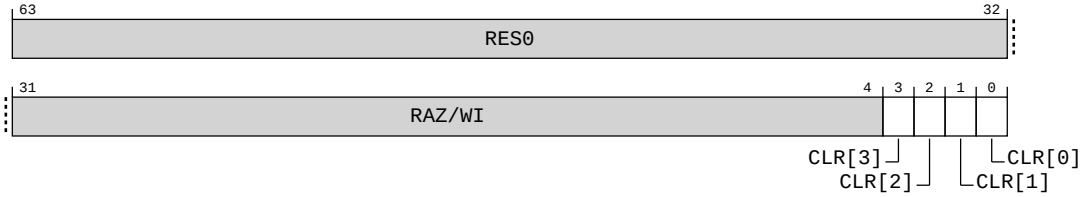


Table A-377: TRCCLAIMCLR bit descriptions

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
[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

A.11 AArch64 Memory Partitioning and Monitoring registers summary

The summary table provides an overview of **IMPLEMENTATION DEFINED** Memory Partitioning and Monitoring registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table A-380: Memory Partitioning and Monitoring registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
MPAMVPMV_EL2	3	4	C10	C4	1	–	64-bit	MPAM Virtual Partition Mapping Valid Register
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

A.11.1 MPAMVPMV_EL2, MPAM Virtual Partition Mapping Valid Register

Valid bits for virtual PARTID mapping entries. Each bit m corresponds to virtual PARTID mapping entry m in the MPAMVPM<n>_EL2 registers where n = m >> 2.

Configurations

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

Attributes

Width

64

Functional group

Memory Partitioning and Monitoring 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-161: AArch64_mpamvpmv_el2 bit assignments

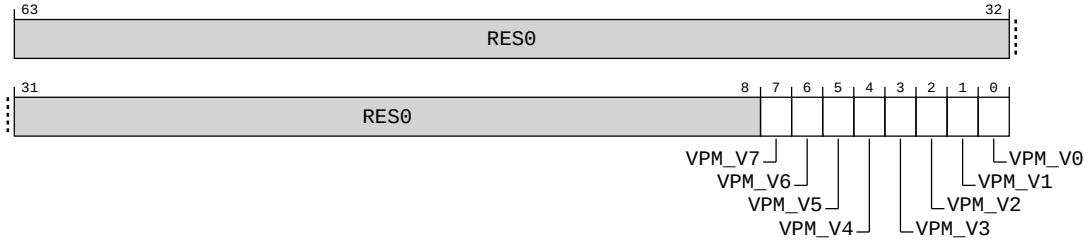


Table A-381: MPAMVPMV_EL2 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7]	VPM_V7	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[6]	VPM_V6	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[5]	VPM_V5	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[4]	VPM_V4	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[3]	VPM_V3	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[2]	VPM_V2	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[1]	VPM_V1	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[0]	VPM_V0	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x

Access

MRS <Xt>, MPAMVPMV_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0100	0b001

MSR MPAMVPMV_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0100	0b001

Accessibility

MRS <Xt>, MPAMVPMV_EL2

```
if PSTATE.EL == EL0 then
```

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
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 MPAMVPMV_EL2;
elsif PSTATE.EL == EL3 then
    return MPAMVPMV_EL2;
```

MSR MPAMVPMV_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPMV_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPMV_EL2 = X[t];
```

A.11.2 MPAMVPM0_EL2, MPAM Virtual PARTID Mapping Register 0

MPAMVPM0_EL2 provides mappings from virtual PARTIDs 0 - 3 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 register. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2.

Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1.

A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

Memory Partitioning and Monitoring 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-162: AArch64_mpamvpm0_el2 bit assignments

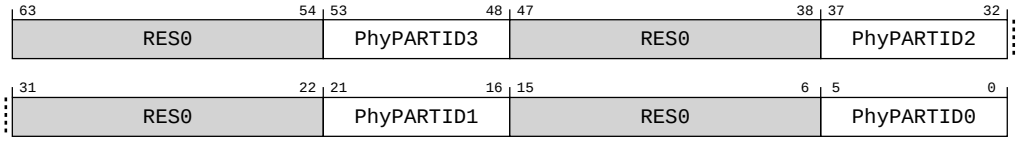


Table A-384: MPAMVPM0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:54]	**RES0**	Reserved	**RES0**
[53:48]	PhyPARTID3	Virtual PARTID Mapping Entry for virtual PARTID 3. PhyPARTID3 gives the mapping of virtual PARTID 3 to a physical PARTID.	6{x}
[47:38]	**RES0**	Reserved	**RES0**
[37:32]	PhyPARTID2	Virtual PARTID Mapping Entry for virtual PARTID 2. PhyPARTID2 gives the mapping of virtual PARTID 2 to a physical PARTID.	6{x}
[31:22]	**RES0**	Reserved	**RES0**
[21:16]	PhyPARTID1	Virtual PARTID Mapping Entry for virtual PARTID 1. PhyPARTID1 gives the mapping of virtual PARTID 1 to a physical PARTID.	6{x}
[15:6]	**RES0**	Reserved	**RES0**
[5:0]	PhyPARTID0	Virtual PARTID Mapping Entry for virtual PARTID 0. PhyPARTID0 gives the mapping of virtual PARTID 0 to a physical PARTID.	6{x}

Access

MRS <Xt>, MPAMVPM0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b000

MSR MPAMVPM0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b000

Accessibility

MRS <Xt>, MPAMVPM0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
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 MPAMVPM0_EL2;
elsif PSTATE.EL == EL3 then
    return MPAMVPM0_EL2;
```

MSR MPAMVPM0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM0_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPM0_EL2 = X[t];
```

A.11.3 MPAMVPM1_EL2, MPAM Virtual PARTID Mapping Register 1

MPAMVPM1_EL2 provides mappings from virtual PARTIDs 4 - 7 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented AArch64-MPAMVPM0_EL2 to AArch64-MPAMVPM7_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2.

Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1.

A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

Memory Partitioning and Monitoring 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-163: AArch64_mpamvpm1_el2 bit assignments

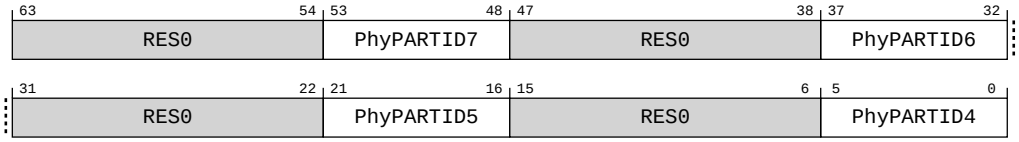


Table A-387: MPAMVPM1_EL2 bit descriptions

Bits	Name	Description	Reset
[63:54]	**RES0**	Reserved	**RES0**
[53:48]	PhyPARTID7	Virtual PARTID Mapping Entry for virtual PARTID 7. PhyPARTID7 gives the mapping of virtual PARTID 7 to a physical PARTID.	6{x}
[47:38]	**RES0**	Reserved	**RES0**
[37:32]	PhyPARTID6	Virtual PARTID Mapping Entry for virtual PARTID 6. PhyPARTID6 gives the mapping of virtual PARTID 6 to a physical PARTID.	6{x}
[31:22]	**RES0**	Reserved	**RES0**
[21:16]	PhyPARTID5	Virtual PARTID Mapping Entry for virtual PARTID 5. PhyPARTID5 gives the mapping of virtual PARTID 5 to a physical PARTID.	6{x}
[15:6]	**RES0**	Reserved	**RES0**
[5:0]	PhyPARTID4	Virtual PARTID Mapping Entry for virtual PARTID 4. PhyPARTID4 gives the mapping of virtual PARTID 4 to a physical PARTID.	6{x}

Access

MRS <Xt>, MPAMVPM1_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b001

MSR MPAMVPM1_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b001

Accessibility

MRS <Xt>, MPAMVPM1_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
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 MPAMVPM1_EL2;
elsif PSTATE.EL == EL3 then
    return MPAMVPM1_EL2;
```

MSR MPAMVPM1_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM1_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPM1_EL2 = X[t];
```

Appendix B External registers

This appendix contains the descriptions for the Cortex®-A715 core 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 **IMPLEMENTATION DEFINED** memory-mapped MPMM registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table B-1: MPMM registers summary

Offset	Name	Reset	Width	Description
0x000	CPUPPMCR	–	32-bit	Global PPM Configuration Register
0x010	CPUMPMMCR	–	32-bit	Global MPMM Configuration Register
0x020	CPUPPMPDPCR [31:0]	–	32-bit	Global PPMPDP Configuration Register
0x024	CPUPPMPDPCR [63:32]	–	32-bit	Global PPMPDP 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 [31:0] are architecturally mapped to AArch64 System register A.2.1 IMP_CPUPPMCR_EL3, Global PPM Configuration Register on page 253.

Attributes

Width

32

Component

MPMM

Register offset

0x000

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-1: ext_cpuppmcr bit assignments

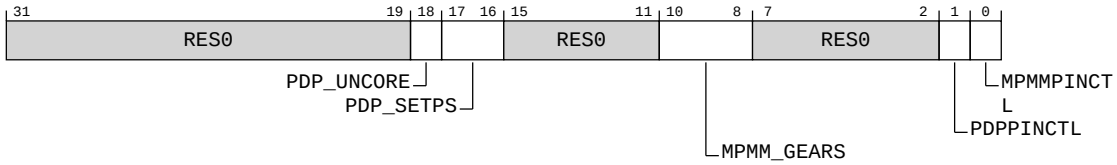


Table B-2: CPUPPMCR bit descriptions

Bits	Name	Description	Reset
[31:19]	**RES0**	Reserved	**RES0**
[18]	PDP_UNCORE	Indicates whether PDP uncore is implemented **0b1** PDP has separate uncore and core controls. Access to this field is: RO	x
[17:16]	PDP_SETPS	Number of PDP Setpoints implemented **0b11** 3 PDP are 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
[7:2]	**RES0**	Reserved	**RES0**
[1]	PDPPINCTL	PDP Pin Control Enabled **0b0** PDP control through SPR and utility bus **0b1** PDP control through pin only.	0b0

Bits	Name	Description	Reset
[0]	MPMMPINCTL	MPMM Pin Control Enabled **0b0** MPMM control through SPR and utility bus. **0b1** MPMM control through pin only.	0b0

Accessibility

Component	Offset	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 [31:0] are architecturally mapped to AArch64 System register A.4.27 IMP_CPUMPMMCR_EL3, Global MPMM Configuration Register on page 320.

Attributes

Width
32

Component
MPMM

Register offset
0x010

Access type
See bit descriptions

Reset value

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

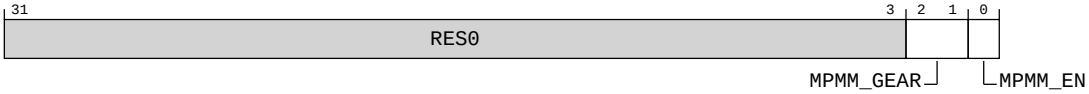


Table B-4: CPUMPMMCR bit descriptions

Bits	Name	Description	Reset
[31:3]	**RES0**	Reserved	**RES0**
[2:1]	MPMM_GEAR	MPMM Gear Select **0b00** Select MPMM Gear 0. **0b01** Select MPMM Gear 1. **0b10** Select MPMM Gear 2. **0b11** Select MPMM Gear 3.	0b00
[0]	MPMM_EN	MPMM Master Enable **0b0** MPMM is not enabled. **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.1.3 CPUPPMPDPCR, Global PPMPDP Configuration Register

This register controls the aggressiveness of PDP features.

Configurations

External register CPUPPMPDPCR bits [63:0] are architecturally mapped to AArch64 System register A.4.26 IMP_CPUPPMPDPCR_EL1, Global PPMPDP Configuration Register on page 318.

Attributes

Width

64

Component

MPMM

Register offsets (2)

0x020,0x024

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-3: ext_cpuppmpdpcr bit assignments

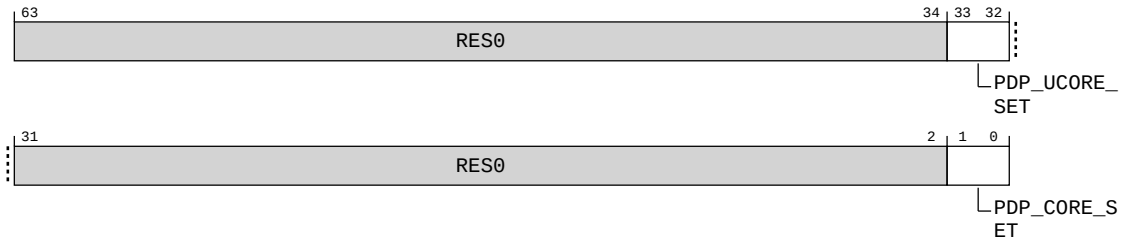


Table B-6: CPUPPMPDPCR bit descriptions

Bits	Name	Description	Reset
[63:34]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[33:32]	PDP_UCORE_SET	Uncore PDP Aggressiveness **0b00** Disable PDP. **0b01** Engage PDP at low aggressiveness **0b10** Engage PDP at medium aggressiveness **0b11** Engage PDP at high aggressiveness	0b00
[31:2]	**RES0**	Reserved	**RES0**
[1:0]	PDP_CORE_SET	Core PDP Aggressiveness **0b00** Disable PDP. **0b01** Engage PDP at low aggressiveness. **0b10** Engage PDP at medium aggressiveness. **0b11** Engage PDP at high aggressiveness.	0b00

Accessibility

Component	Offset	Instance	Range
MPMM	0x020	CPUPPMPDPCR	31:0

This interface is accessible as follows:

When IsCorePowered() && IsAccessSecure()
RW

When IsCorePowered() && !IsAccessSecure()
RAZ/WI

Otherwise
ERROR

Component	Offset	Instance	Range
MPMM	0x024	CPUPPMPDPCR	63:32

This interface is accessible as follows:

When IsCorePowered() && IsAccessSecure()
RW

When IsCorePowered() && !IsAccessSecure()
RAZ/WI

Otherwise

ERROR

B.2 External PMU registers summary

The summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped PMU registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table B-9: PMU registers summary

Offset	Name	Reset	Width	Description
0x600	PMPCSSR	—	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	—	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x60C	PMCID2SSR	—	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register
0x610	PMSSSR	—	32-bit	PMU Snapshot Status 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
0x6F0	PMSSCR	—	32-bit	PMU Snapshot Capture Register
0xE00	PMCFGR	—	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_EL0	—	32-bit	Performance Monitors Control Register

Offset	Name	Reset	Width	Description
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
0xE40	PMMIR	—	32-bit	Performance Monitors Machine Identification 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.2.1 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-4: ext_pmpcssr bit assignments

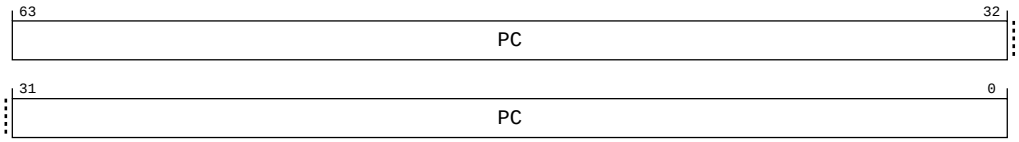


Table B-10: PMPCSSR bit descriptions

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}

B.2.2 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-5: ext_pmcidssr bit assignments

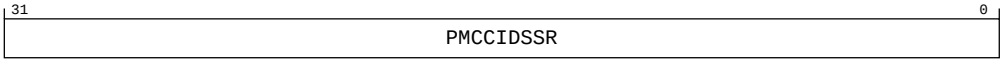


Table B-11: PMCIDSSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMCCIDSSR	PMCIDSR sample. Sampled CONTEXTIDR_EL1 snapshot.	32{x}

B.2.3 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register

Captured copy of the CONTEXTIDR_EL2 register.

The value captured must relate to the instruction captured in PMPCSSR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x60C

Access type

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-6: ext_pmcid2ssr bit assignments

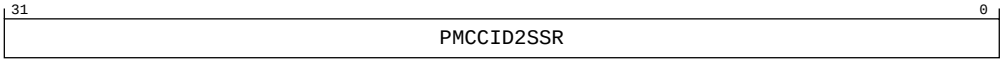


Table B-12: PMCID2SSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMCCID2SSR	PMCID2SR sample. Sampled CONTEXTIDR_EL2 snapshot.	32{x}

B.2.4 PMSSSR, PMU Snapshot Status Register

Holds status information about the captured counters.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x610

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx1

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-7: ext_pmsssr bit assignments

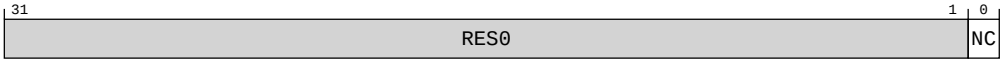


Table B-13: PMSSSR bit descriptions

Bits	Name	Description	Reset
[31:1]	**RES0**	Reserved	**RES0**
[0]	NC	No capture. Indicates whether the PMU counters have been captured. **0b0** PMU counters captured. **0b1** PMU counters not captured. 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

B.2.5 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-8: ext_pmccntsr bit assignments

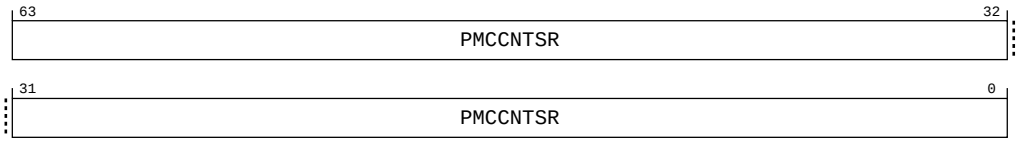


Table B-14: PMCCNTSR bit descriptions

Bits	Name	Description	Reset
[63:0]	PMCCNTSR	PMCCNTR_EL0 sample. Sampled cycle count.	64{x}

B.2.6 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-9: ext_pmevcntsr0 bit assignments

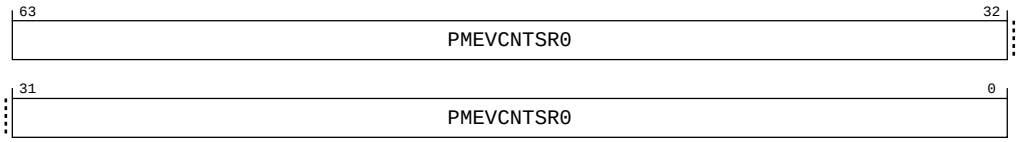


Table B-15: PMEVCNTSR0 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR0	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.7 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-10: ext_pmevcntsr1 bit assignments

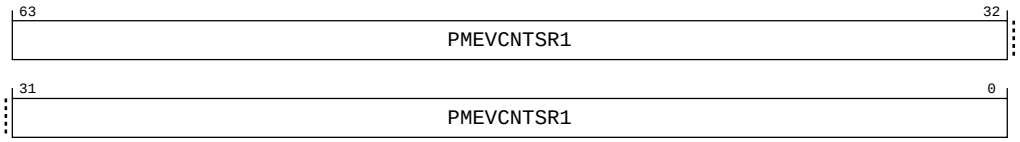


Table B-16: PMEVCNTSR1 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR1	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.8 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-11: ext_pmevcntsr2 bit assignments

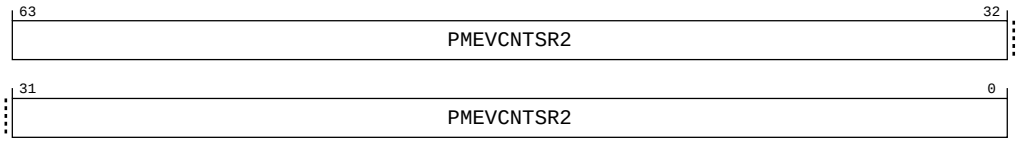


Table B-17: PMEVCNTSR2 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR2	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.9 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-12: ext_pmevcntsr3 bit assignments

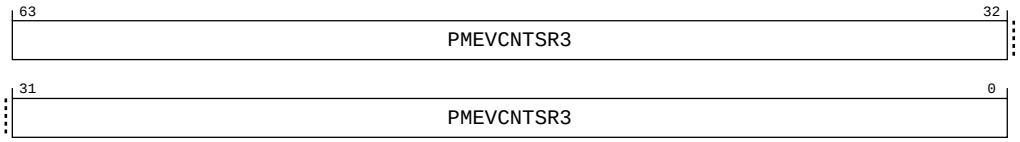


Table B-18: PMEVCNTSR3 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR3	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.10 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-13: ext_pmevcntsr4 bit assignments

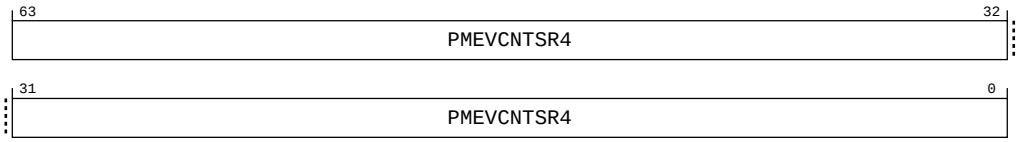


Table B-19: PMEVCNTSR4 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR4	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.11 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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-14: ext_pmevcntsr5 bit assignments

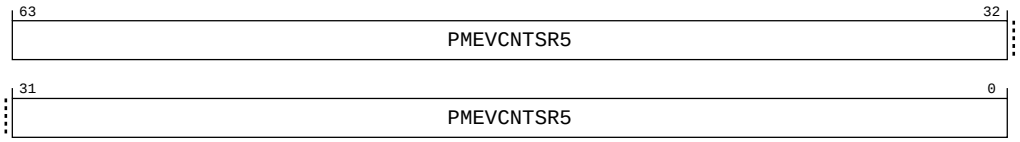


Table B-20: PMEVCNTSR5 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR5	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.12 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-15: ext_pmevcntsr6 bit assignments

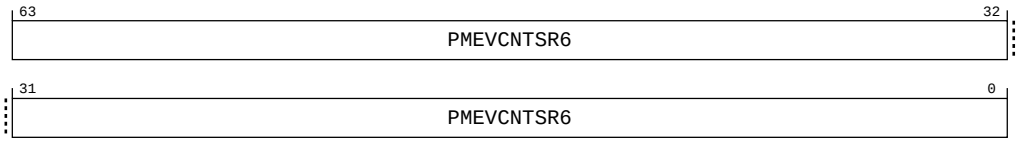


Table B-21: PMEVCNTSR6 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR6	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.13 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-16: ext_pmevcntsr7 bit assignments

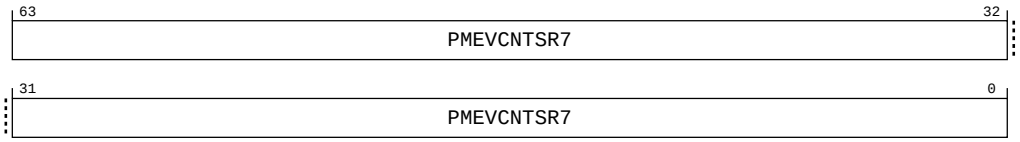


Table B-22: PMEVCNTSR7 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR7	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.14 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-17: ext_pmevcntsr8 bit assignments

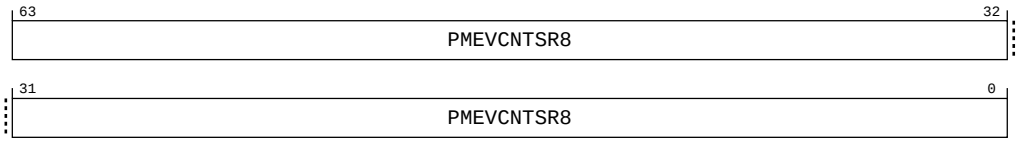


Table B-23: PMEVCNTSR8 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR8	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.15 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-18: ext_pmevcntsr9 bit assignments

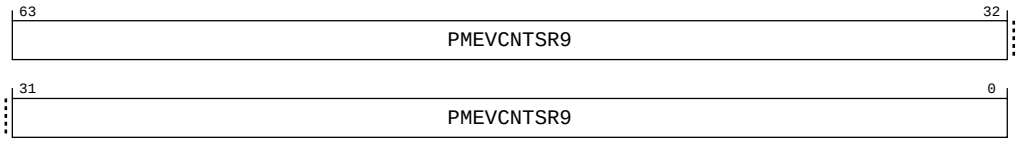


Table B-24: PMEVCNTSR9 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR9	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.16 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-19: ext_pmevcntsr10 bit assignments

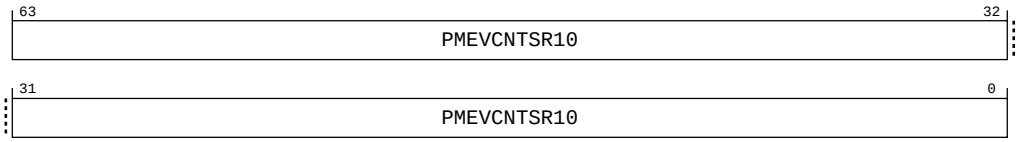


Table B-25: PMEVCNTSR10 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR10	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.17 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-20: ext_pmevcntsr11 bit assignments

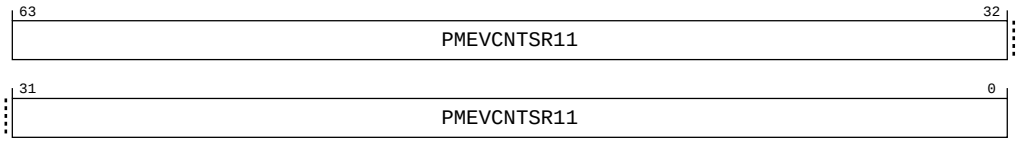


Table B-26: PMEVCNTSR11 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR11	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.18 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-21: ext_pmevcntsr12 bit assignments

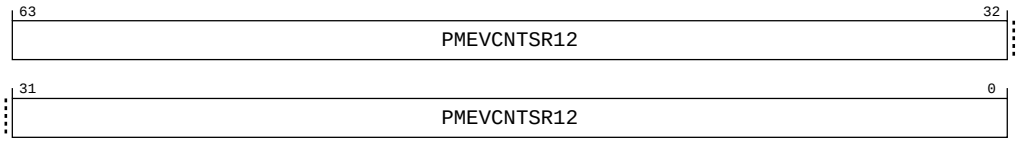


Table B-27: PMEVCNTSR12 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR12	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.19 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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-22: ext_pmevcntsr13 bit assignments

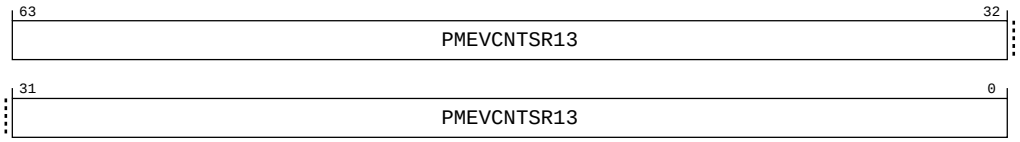


Table B-28: PMEVCNTSR13 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR13	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.20 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-23: ext_pmevcntsr14 bit assignments

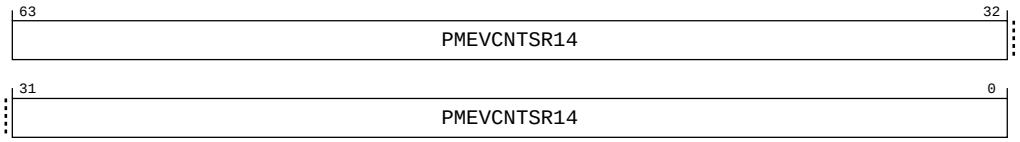


Table B-29: PMEVCNTSR14 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR14	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.21 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-24: ext_pmevcntsr15 bit assignments

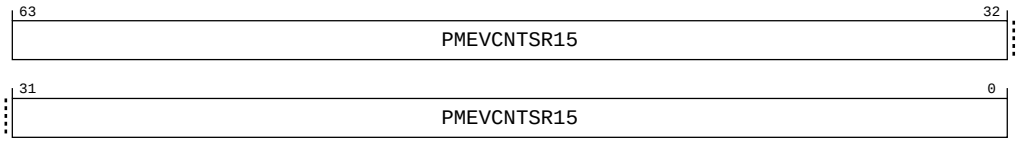


Table B-30: PMEVCNTSR15 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR15	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.22 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-25: ext_pmevcntsr16 bit assignments

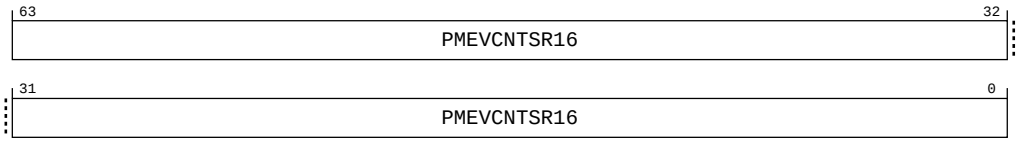


Table B-31: PMEVCNTSR16 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR16	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.23 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-26: ext_pmevcntsr17 bit assignments

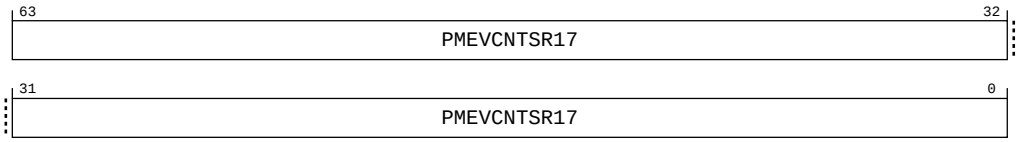


Table B-32: PMEVCNTSR17 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR17	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.24 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-27: ext_pmevcntsr18 bit assignments

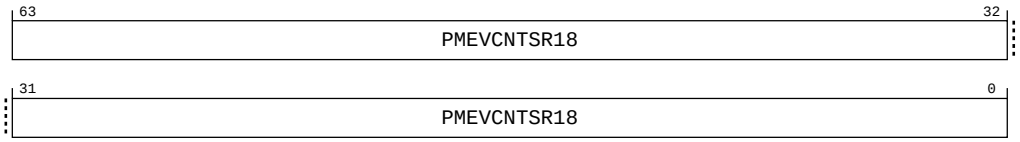


Table B-33: PMEVCNTSR18 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR18	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.25 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-28: ext_pmevcntsr19 bit assignments

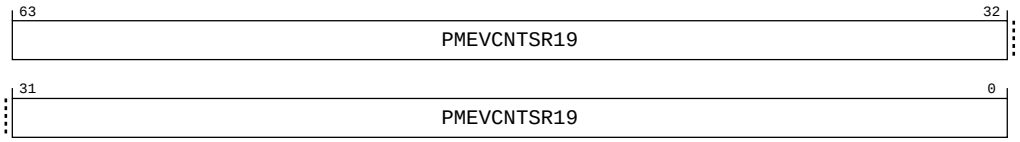


Table B-34: PMEVCNTSR19 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR19	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

B.2.26 PMSSCR, PMU Snapshot Capture Register

Provides a mechanism for software to initiate a sample.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x6F0

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-29: ext_pmsscr bit assignments

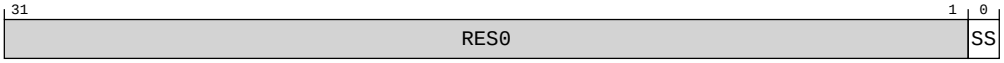


Table B-35: PMSSCR bit descriptions

Bits	Name	Description	Reset
[31:1]	**RES0**	Reserved	**RES0**
[0]	SS	Capture now. **0b0** Ignored. **0b1** Initiate a capture immediately.	x

B.2.27 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

xxxx xxxx xxxx xxx0 xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-30: ext_pmcfgr bit assignments

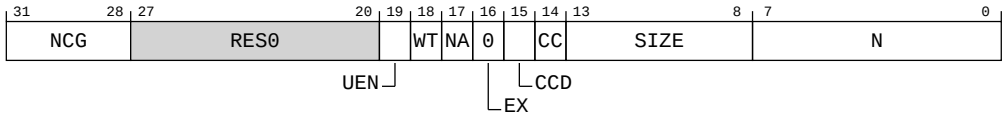


Table B-36: PMCFGR bit descriptions

Bits	Name	Description	Reset
[31:28]	NCG	This feature is not supported, so this field is **RAZ**.	xxxx
[27:20]	**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**.	x
[18]	WT	This feature is not supported, so this bit is **RAZ**.	x
[17]	NA	This feature is not supported, so this bit is **RAZ**.	x

Bits	Name	Description	Reset
[16]	EX	Export supported. Value is **IMPLEMENTATION DEFINED**. **0b0** Export Not supported	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. This bit is **RAO**.	x
[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.	6{x}
[7:0]	N	Number of counters implemented in addition to the cycle counter, ext-PMCCNTR_EL0. The maximum number of event counters is 31. **0b00000000** Only ext-PMCCNTR_EL0 implemented. **0b00000001** ext-PMCCNTR_EL0 plus one event counter implemented.	8{x}

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.2.28 PMCR_EL0, External 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.5.2 PMCR_EL0, Performance Monitors Control Register on page 323.

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-31: ext_pmcr_el0 bit assignments

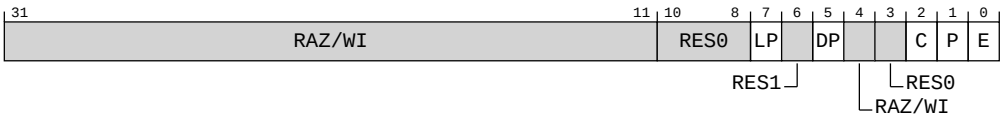


Table B-38: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[31:11]	RAZ/WI	Reserved	RAZ/WI
[10:8]	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. **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 EL2 is implemented and AArch64-MDCR_EL2.HPMN is less than PMCR_EL0.N, this bit does not affect the operation of event counters in the range [AArch64-MDCR_EL2.HPMN:(PMCR_EL0.N-1)]. **Note:** The effect of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state. For more information, see the description of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN.	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 bit. **0b1** When event counting for counters in the range [0..(AArch64-MDCR_EL2.HPMN-1)] is prohibited, cycle counting by ext-PMCCNTR_EL0 is disabled. For more information, see *Prohibiting event counting* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x [2]
[4]	**RAZ/WI**	Reserved	**RAZ/WI**
[3]	**RES0**	Reserved	**RES0**
[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

[2] When this register has an architecturally-defined reset value, if this field is implemented as an RW field it resets to:
- A value that is architecturally **UNKNOWN** if the reset is into an Exception level that is using AArch64.

Bits	Name	Description	Reset
[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. **0b0** All event counters in the range [0..(PMN-1)] and ext-PMCCNTR_EL0, are disabled. **0b1** All event counters in the range [0..(PMN-1)] and ext-PMCCNTR_EL0, are enabled by ext-PMCNTENSET_EL0. If EL2 is implemented then: • If EL2 is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. • If PMN is less than PMCR_EL0.N, this bit does not affect the operation of event counters in the range [PMN..(PMCR_EL0.N-1)]. If EL2 is not implemented, PMN is PMCR_EL0.N. **Note:** The effect of the following fields on the operation of this bit applies if EL2 is implemented regardless of whether EL2 is enabled in the current Security state: • AArch64-MDCR_EL2.HPMN. See the description of AArch64-MDCR_EL2.HPMN for more information.	0b0

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() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.2.29 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.5.3 PMCEID0_EL0, Performance Monitors Common Event Identification register 0 on page 327.

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-32: ext_pmceid0 bit assignments

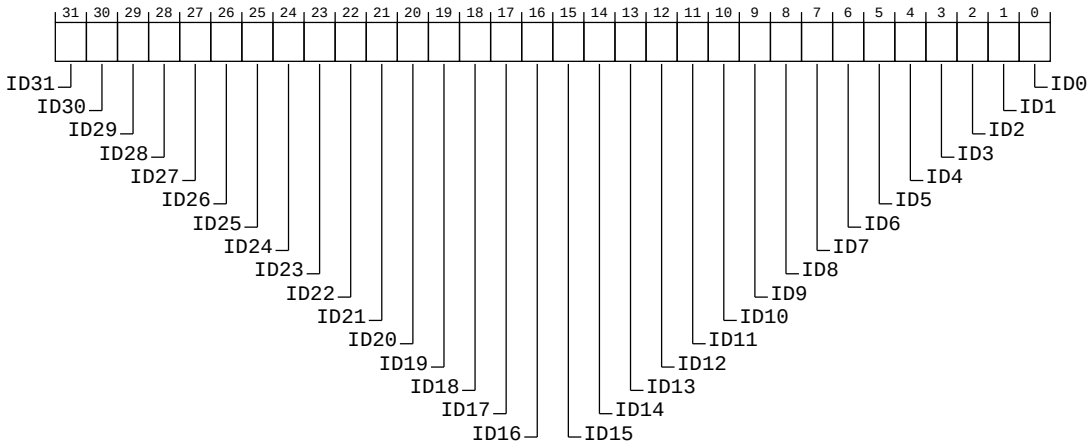


Table B-40: PMCEID0 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x1f) L1D_CACHE_ALLOCATE **0b0** The common event is not implemented, or not counted.	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 **0b0** The common event is not implemented, or not counted.	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 **0b1** The common event is implemented.	x

Bits	Name	Description	Reset
[23]	ID23	ID23 corresponds to common event (0x17) L2D_CACHE_REFILL **0b1** The common event is implemented.	x
[22]	ID22	ID22 corresponds to common event (0x16) L2D_CACHE **0b1** The common event is implemented.	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
[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

Bits	Name	Description	Reset
[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 **0b0** The common event is not implemented, or not counted.	x
[6]	ID6	ID6 corresponds to common event (0x6) LD_RETIRED **0b0** The common event is not implemented, or not counted.	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

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.2.30 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.5.4 PMCEID1_EL0, Performance Monitors Common Event Identification register 1 on page 334.

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-33: ext_pmceid1 bit assignments

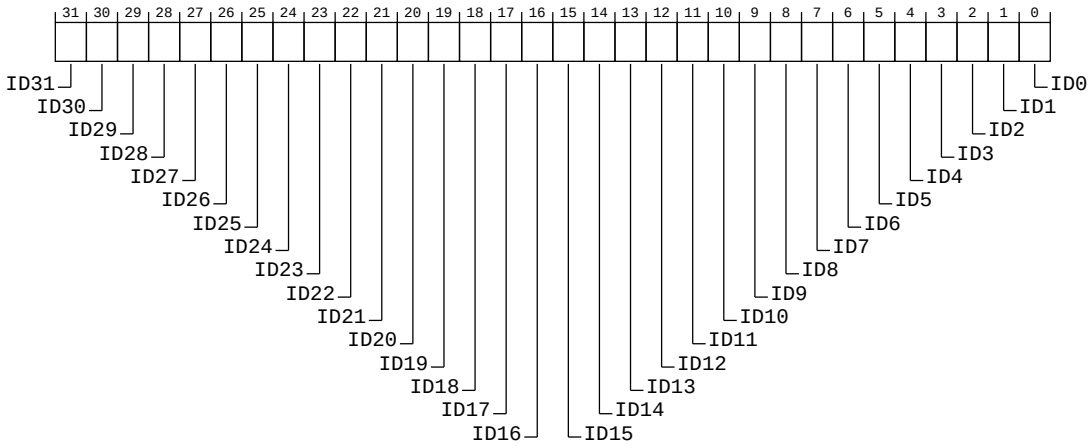


Table B-42: PMCEID1 bit descriptions

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 **0b0** The common event is not implemented, or not counted.	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 **0b1** The common event is implemented.	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 **0b1** The common event is implemented.	x
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL **0b1** The common event is implemented.	x

Bits	Name	Description	Reset
[9]	ID9	ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE **0b1** The common event is implemented.	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
[0]	ID0	ID0 corresponds to common event (0x20) L2D_CACHE_ALLOCATE **0b1** The common event is implemented.	x

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.2.31 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 [63:32] are architecturally mapped to AArch64 System register A.5.3 PMCEID0_EL0, Performance Monitors Common Event Identification register 0 on page 327.

Attributes

Width

32

Component

PMU

Register offset

0xE28

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-34: ext_pmceid2 bit assignments

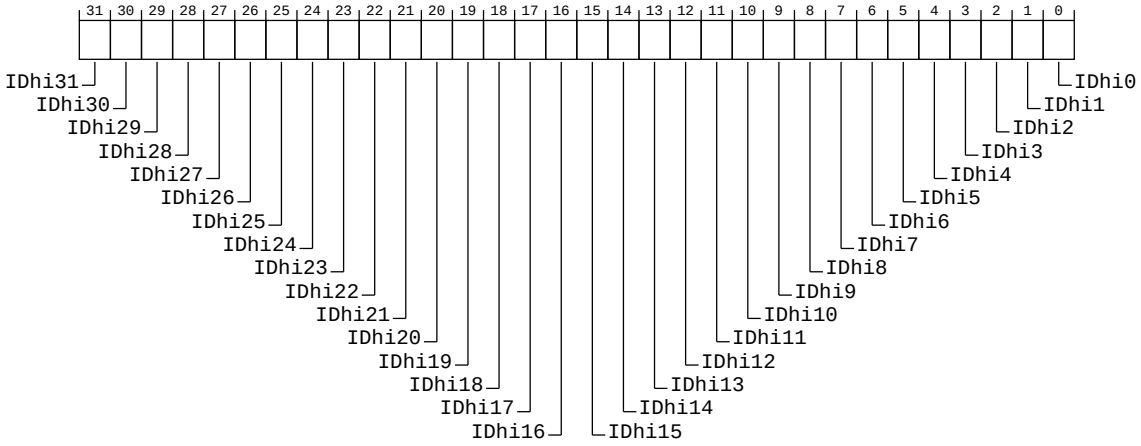


Table B-44: PMCEID2 bit descriptions

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
[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

Bits	Name	Description	Reset
[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
[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 **0b0** The common event is not implemented, or not counted.	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 **0b1** The common event is implemented.	x
[10]	IDhi10	IDhi10 corresponds to common event (0x400a) L2I_CACHE_LMISS **0b0** The common event is not implemented, or not counted.	x

Bits	Name	Description	Reset
[9]	IDhi9	IDhi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD **0b1** The common event is implemented.	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
[4]	IDhi4	IDhi4 corresponds to common event (0x4004) CNT_CYCLES **0b1** The common event is implemented.	x
[3]	IDhi3	IDhi3 corresponds to common event (0x4003) SAMPLE_COLLISION **0b0** The common event is not implemented, or not counted. This value is reported if the SPE is not implemented. **0b1** The common event is implemented. This value is reported if SPE is implemented.	x
[2]	IDhi2	IDhi2 corresponds to common event (0x4002) SAMPLE_FILTRATE **0b0** The common event is not implemented, or not counted. This value is reported if the SPE is not implemented. **0b1** The common event is implemented. This value is reported if SPE is implemented.	x
[1]	IDhi1	IDhi1 corresponds to common event (0x4001) SAMPLE_FEED **0b0** The common event is not implemented, or not counted. This value is reported if the SPE is not implemented. **0b1** The common event is implemented. This value is reported if SPE is implemented.	x
[0]	IDhi0	IDhi0 corresponds to common event (0x4000) SAMPLE_POP **0b0** The common event is not implemented, or not counted. This value is reported if the SPE is not implemented. **0b1** The common event is implemented. This value is reported if SPE is implemented.	x

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.2.32 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 [63:32] are architecturally mapped to AArch64 System register A.5.4 PMCEID1_EL0, Performance Monitors Common Event Identification register 1 on page 334.

Attributes

Width

32

Component

PMU

Register offset

0xE2C

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-35: ext_pmceid3 bit assignments

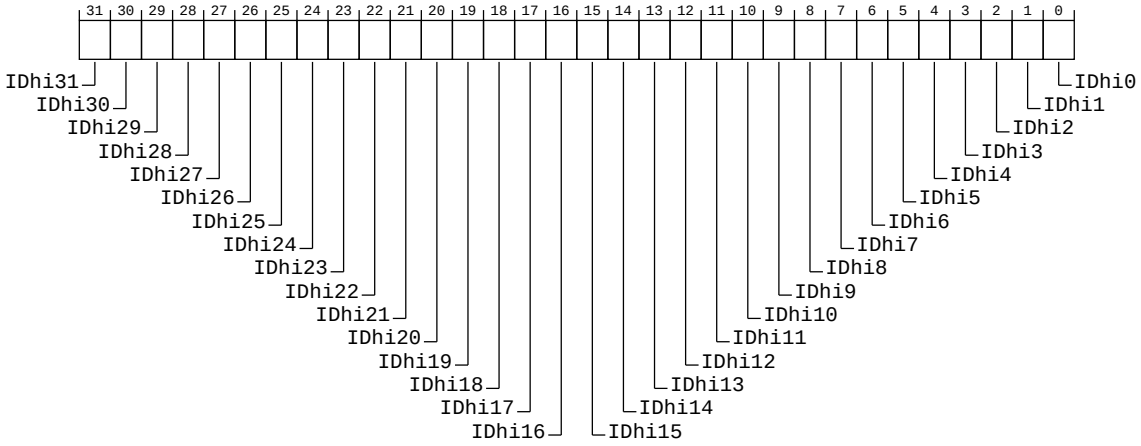


Table B-46: PMCEID3 bit descriptions

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_CHECKED_WR **0b1** The common event is implemented.	x
[5]	IDhi5	IDhi5 corresponds to common event (0x4025) MEM_ACCESS_CHECKED_RD **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

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.2.33 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 xxxx xxxx xxxx 0000 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-36: ext_pmmir bit assignments

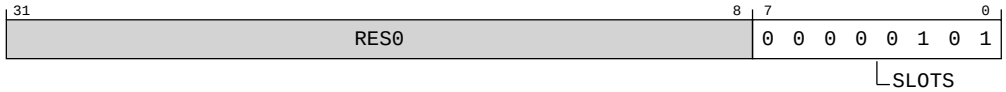


Table B-48: PMMIR bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is implemented, this field must not be zero. **0b00000101** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 5.	0x05

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.2.34 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

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-37: ext_pmdevarch bit assignments

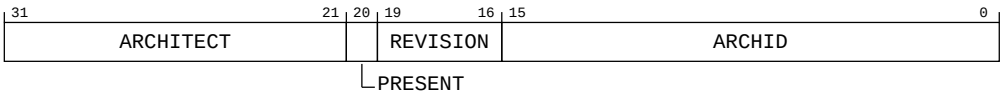


Table B-50: PMDEVARCH bit descriptions

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`.	`11{x}`
[20]	PRESENT	When set to 1, indicates that the DEVARCH is present. This field is 1 in Armv8.	`x`
[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.	`xxxx`
[15:0]	ARCHID	Defines this part to be an Armv8 debug component. For architectures defined by Arm this is further subdivided. For Performance Monitors: • Bits [15:12] are the architecture version, `0x2`. • Bits [11:0] are the architecture part number, `0xA16`. This corresponds to Performance Monitors architecture version PMUv3. **Note:** The PMUv3 memory-mapped programmers' model can be used by devices other than Armv8 processors. Software must determine whether the PMU is attached to an Armv8 processor by using the ext-PMDEVAFF0 and ext-PMDEVAFF1 registers to discover the affinity of the PMU to any Armv8 processors.	`16{x}`

Accessibility

Component	Offset	Instance	Range
PMU	0xFBC	PMDEVARCH	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.35 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.

Note

Before Armv8.2, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of 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 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-38: ext_pmdevid bit assignments

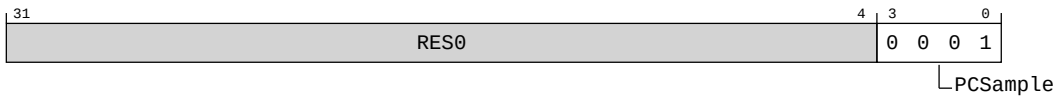


Table B-52: PMDEVID bit descriptions

Bits	Name	Description	Reset
[31:4]	**RES0**	Reserved	**RES0**
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using Performance Monitors registers. **0b0001** PC Sample-based Profiling Extension is implemented in the Performance Monitors register space.	0b0001

Accessibility

Component	Offset	Instance	Range
PMU	0xFC8	PMDEVID	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.2.36 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 XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-39: ext_pmdevtype bit assignments

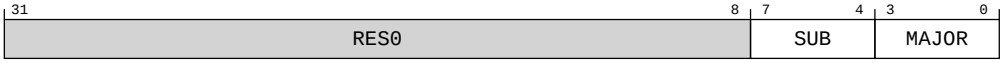


Table B-54: PMDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Subtype. Must read as `0x1` to indicate this is a component within a PE.	xxxx
[3:0]	MAJOR	Major type. Must read as `0x6` to indicate this is a performance monitor component.	xxxx

Accessibility

Component	Offset	Instance	Range
PMU	0xFCC	PMDEVTYPE	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.37 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 xxxx 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-40: ext_pmpidr4 bit assignments

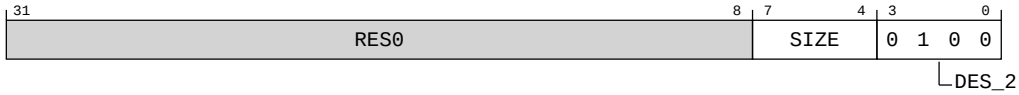


Table B-56: PMPIDR4 bit descriptions

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.	xxxx
[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.2.38 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 0100 1101

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-41: ext_pmpidr0 bit assignments

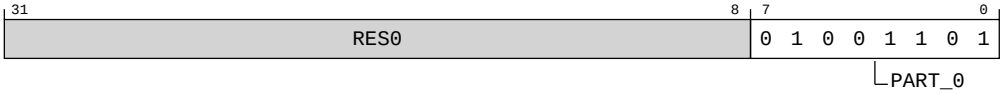


Table B-58: PMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b01001101** A715	0x4D

Accessibility

Component	Offset	Instance	Range
PMU	0xFE0	PMPIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.39 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-42: ext_pmpidr1 bit assignments

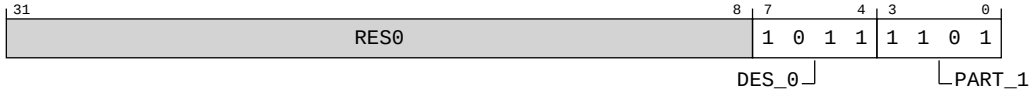


Table B-60: PMPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is `0b1011`. **`0b1011`** Arm Limited	`0b1011`
[3:0]	PART_1	Part number, most significant nibble. **`0b1101`** A715	`0b1101`

Accessibility

Component	Offset	Instance	Range
PMU	0xFE4	PMPIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.40 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 0001 x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-43: ext_pmpidr2 bit assignments

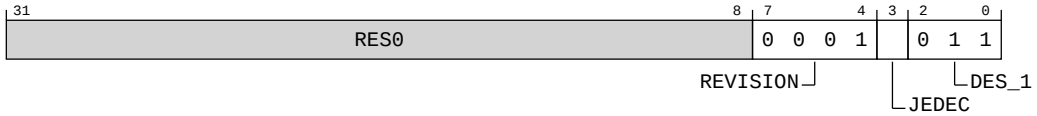


Table B-62: PMPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0001** r1p2	0b0001
[3]	JEDEC	**RAO**. Indicates a JEP106 identity code is used.	x
[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.2.41 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-44: ext_pmpidr3 bit assignments

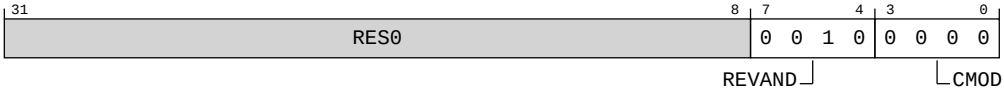


Table B-64: PMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[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** r1p2	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.2.42 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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-45: ext_pmcidr0 bit assignments

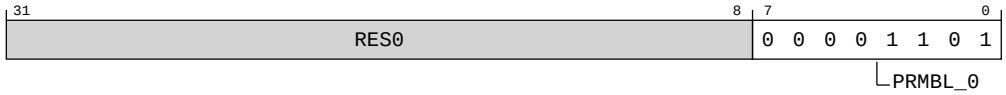


Table B-66: PMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble.	0x0D

Accessibility

Component	Offset	Instance	Range
PMU	0xFF0	PMCIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.43 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 1001 0000

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-46: ext_pmcidr1 bit assignments

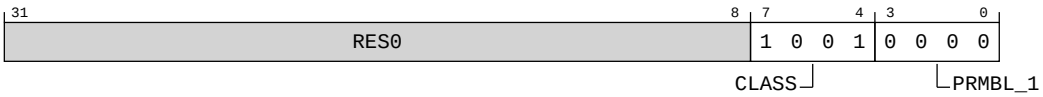


Table B-68: PMCIDR1 bit descriptions

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.	0b1001
[3:0]	PRMBL_1	Preamble. **RAZ**.	0b0000

Accessibility

Component	Offset	Instance	Range
PMU	0xFF4	PMCIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.44 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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-47: ext_pmcidr2 bit assignments

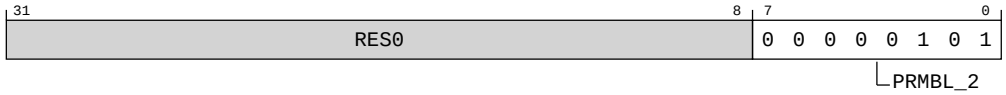


Table B-70: PMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble.	0x05

Accessibility

Component	Offset	Instance	Range
PMU	0xFF8	PMCIDR2	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.45 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-48: ext_pmcidr3 bit assignments

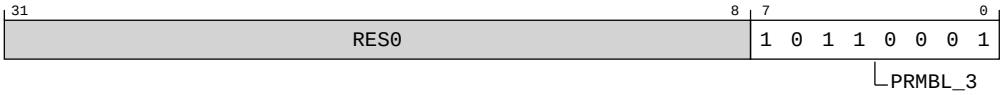


Table B-72: PMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble.	0xB1

Accessibility

Component	Offset	Instance	Range
PMU	0xFFC	PMCIDR3	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3 External Debug registers summary

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

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table B-74: Debug registers summary

Offset	Name	Reset	Width	Description
0x090	EDRCR	—	32-bit	External Debug Reserve Control Register
0x094	EDACR	—	32-bit	External Debug Auxiliary Control Register
0x310	EDPRCR	—	32-bit	External Debug Power/Reset Control Register
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
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

Offset	Name	Reset	Width	Description
0xFFC	EDCIDR3	—	32-bit	External Debug Component Identification Register 3

B.3.1 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-49: ext_edrcr bit assignments

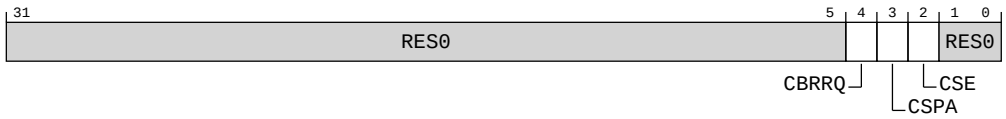


Table B-75: EDRCR bit descriptions

Bits	Name	Description	Reset
[31:5]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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. If this feature is not implemented, writes to this bit are ignored.	x
[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() && SoftwareLockStatus()
WI

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && !SoftwareLockStatus()
WO

Otherwise
ERROR

B.3.2 EDACR, External Debug Auxiliary Control Register

Allows implementations to support **IMPLEMENTATION DEFINED** controls.

Configurations

If FEAT_DoPD is implemented, this register is implemented in the Core power domain.

If FEAT_DoPD is not implemented, the power domain that this register is implemented in is IMPLEMENTATION DEFINED.

Changing this register from its reset value causes IMPLEMENTATION DEFINED behavior, including possible deviation from the architecturally-defined behavior.

If the EDACR contains any control bits that must be preserved over power down, then these bits must be accessible by the external debug interface when the OS Lock is locked, AArch64-OSLSR_EL1.OSLK == 1, and when the Core is powered off.

Attributes

Width

32

Component

Debug

Register offset

0x094

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-50: ext_edacr bit assignments

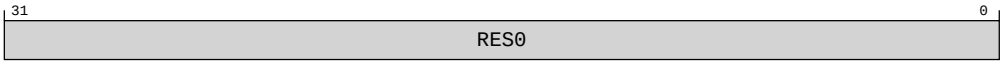


Table B-77: EDACR bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
Debug	0x094	EDACR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && SoftwareLockStatus()
RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && !SoftwareLockStatus()
RW

Otherwise
ImplementationDefined

B.3.3 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 xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-51: ext_edprcr bit assignments

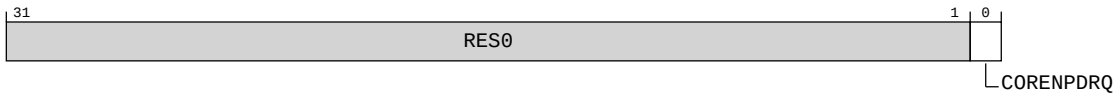


Table B-79: EDPRCR bit descriptions

Bits	Name	Description	Reset
[31:1]	**RES0**	Reserved	**RES0**
[0]	CORENPDRQ	Core no powerdown request. Requests emulation of powerdown. This request is typically passed to an external power controller. This means that whether a request causes power up is dependent on the **IMPLEMENTATION DEFINED** nature of the system. The power controller must not allow the Core power domain to switch off while this bit is 1. **0b0** If the system responds to a powerdown request, it powers down Core power domain. **0b1** If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain. When 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 **When SoftwareLockStatus()** Access to this field is: RO **Otherwise** Access to this field is: RW	x [3]

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:

[3] On a Cold reset, if the powerup request is implemented and the powerup request has been asserted, this field is an **IMPLEMENTATION DEFINED** choice of 0 or 1. If the powerup request is not asserted, this field is set to 0.

When IsCorePowered() && SoftwareLockStatus()

RO

When IsCorePowered() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.4 MIDR_EL1, External 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.4.1 MIDR_EL1, Main ID Register on page 264.

Attributes

Width

32

Component

Debug

Register offset

0xD00

Access type

See bit descriptions

Reset value

0100 0001 0001 1111 1101 0100 1101 0010

Bit descriptions

Figure B-52: ext_midr_el1 bit assignments

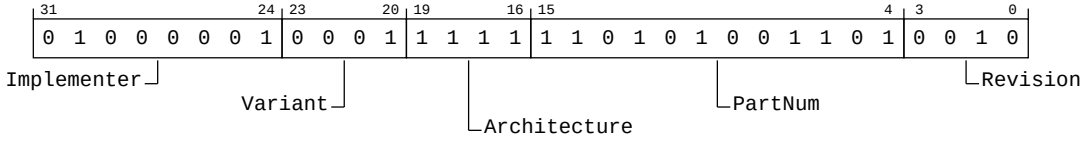


Table B-81: MIDR_EL1 bit descriptions

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. **0b0001** r1p2	0b0001
[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. **0b110101001101** A715	0xD4D
[3:0]	Revision	Indicates the minor revision of the product. **0b0010** r1p2	0b0010

Accessibility

Component	Offset	Instance	Range
Debug	0xD00	MIDR_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

B.3.5 EDPFR, External Debug Processor Feature Register

Provides information about implemented PE features.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offsets (2)

0xD20,0xD24

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx 0001 xxxx 0001 0001 xxxx xxxx 0001 0001 0001 0001 0001 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-53: ext_edpfr bit assignments

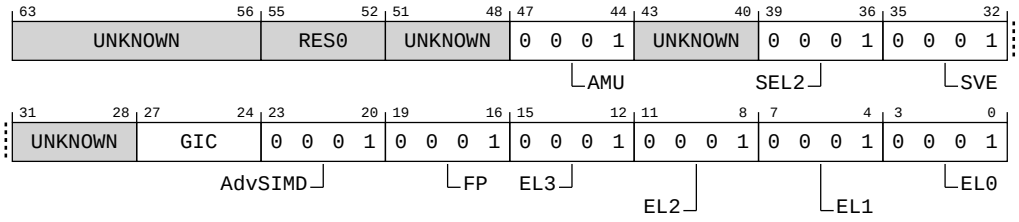


Table B-83: EDPFR bit descriptions

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.	0b0001
[43:40]	UNKNOWN	Reserved	UNKNOWN
[39:36]	SEL2	Secure EL2. Defined values are: **0b0001** Secure EL2 is implemented.	0b0001

Bits	Name	Description	Reset
[35:32]	SVE	Scalable Vector Extension. Defined values are: **0b0001** SVE is implemented.	0b0001
[31:28]	UNKNOWN	Reserved	UNKNOWN
[27:24]	GIC	System register GIC interface support. Defined values are: **0b0000** GIC CPU interface system registers not implemented. **0b0011** System register interface to version 4.1 of the GIC CPU interface is supported.	The reset values can be the following: 0b0000, 0b0011, respective to the value.
[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.	0b0001
[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.	0b0001
[15:12]	EL3	AArch64 EL3 Exception level handling. Defined values are: **0b0001** EL3 can be executed in AArch64 state only.	0b0001
[11:8]	EL2	AArch64 EL2 Exception level handling. Defined values are: **0b0001** EL2 can be executed in AArch64 state only.	0b0001
[7:4]	EL1	AArch64 EL1 Exception level handling. Defined values are: **0b0001** EL1 can be executed in AArch64 state only.	0b0001
[3:0]	EL0	AArch64 EL0 Exception level handling. Defined values are: **0b0001** EL0 can be executed in AArch64 state only.	0b0001

Accessibility

Component	Offset	Instance	Range
Debug	0xD20	EDPFR	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

Component	Offset	Instance	Range
Debug	0xD24	EDPFR	63:32

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

B.3.6 EDDFR, External Debug Feature Register

Provides top-level information about the debug system.

Debuggers must use ext-EDDEVARCH to determine the Debug architecture version.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* 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 0001 xxxx xxxx 0001 xxxx 0011 xxxx 0101 0110 0001 xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-54: ext_eddfr bit assignments

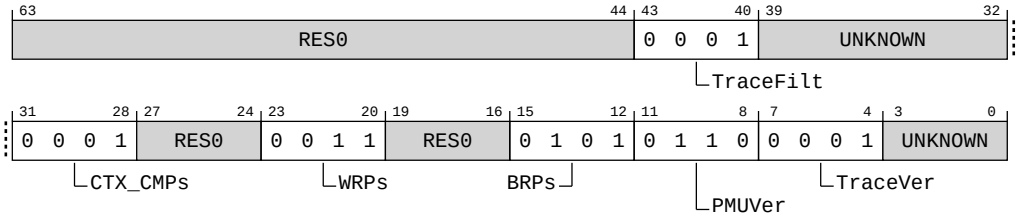


Table B-86: EDDFR bit descriptions

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.	0b0001
[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 state in at least one Exception level, 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 state in at least one Exception level, this field returns the value of AArch64-ID_AA64DFR0_EL1.WRPs. **0b0011** Four watchpoints	0b0011
[19:16]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[15:12]	BRPs	Number of breakpoints, minus 1. The value of `0b0000` is reserved. In an Armv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of AArch64-ID_AA64DFR0_EL1.BRPs. **`0b0101`** Six breakpoints	`0b0101`
[11:8]	PMUVer	Performance Monitors Extension version. Defined value is: **`0b0110`** Performance Monitors Extension implemented, PMUv3 for Armv8.5	`0b0110`
[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.	`0b0001`
[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
ImplementationDefined

Component	Offset	Instance	Range
Debug	0xD2C	EDDFR	63:32

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()
RO

Otherwise
ImplementationDefined

B.3.7 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

xxxx xxxx xxxx xxxx 1001 1010 0001 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-55: ext_eddevarch bit assignments

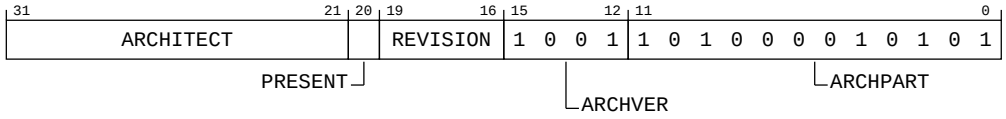


Table B-89: EDDEVARCH bit descriptions

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`.	`11{x}`
[20]	PRESENT	When set to 1, indicates that the DEVARCH is present. This field is 1 in Armv8.	`x`
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. For debug, the revision defined by Armv8-A is `0x0`. All other values are reserved.	`xxxx`

Bits	Name	Description	Reset
[15:12]	ARCHVER	Defines the architecture version of the component. This is the same value as AArch64-ID_AA64DFR0_EL1.DebugVer and AArch32-DBGDIDR.Version. The defined values of this field are: **0b1001** Armv8.4 Debug architecture.	0b1001
[11:0]	ARCHPART	**0b101000010101** The part number of the Armv8-A debug component. The fields ARCHVER and ARCHPART together form the field ARCHID, so that ARCHPART is ARCHID[11:0].	0xA15

Accessibility

Component	Offset	Instance	Range
Debug	0xFBC	EDDEVARCH	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.3.8 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-56: ext_eddevid2 bit assignments

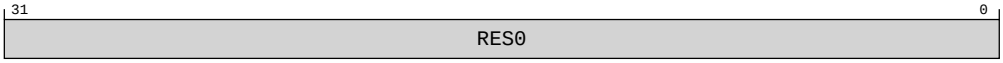


Table B-91: EDDEVID2 bit descriptions

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.3.9 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 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-57: ext_eddevid1 bit assignments

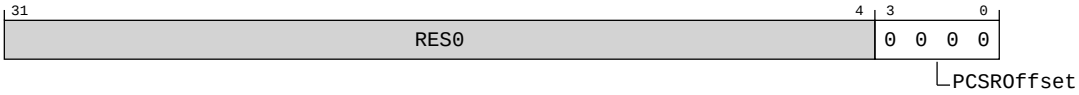


Table B-93: EDDEVID1 bit descriptions

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.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFC4	EDDEVID1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.10 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 0000 xxxx xxxx xxxx xxxx 0001 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-58: ext_eddevid bit assignments

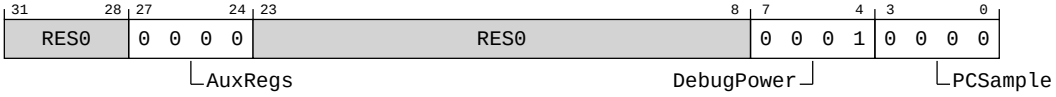


Table B-95: EDDEVID bit descriptions

Bits	Name	Description	Reset
[31:28]	**RES0**	Reserved	**RES0**
[27:24]	AuxRegs	Indicates support for Auxiliary registers. Defined values are: **0b0000** None supported.	0b0000
[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.	0b0001
[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.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFC8	EDDEVID	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.11 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 xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-59: ext_eddevtype bit assignments

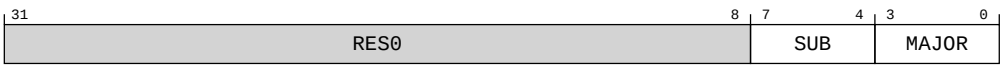


Table B-97: EDDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Subtype. Must read as `0x1` to indicate this is a component within a PE.	xxxx
[3:0]	MAJOR	Major type. Must read as `0x5` to indicate this is a debug logic component.	xxxx

Accessibility

Component	Offset	Instance	Range
Debug	0xFCC	EDDEVTYPE	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.3.12 EDPIDR4, External Debug Peripheral Identification Register 4

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* 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 xxxx 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-60: ext_edpidr4 bit assignments

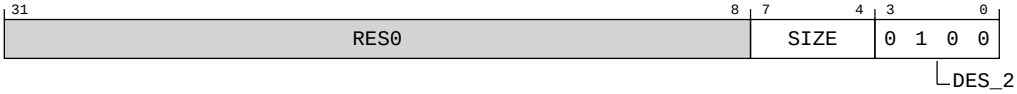


Table B-99: EDPIDR4 bit descriptions

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.	xxxx
[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.3.13 EDPIDR0, External Debug Peripheral Identification Register 0

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* 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 0100 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-61: ext_edpidr0 bit assignments

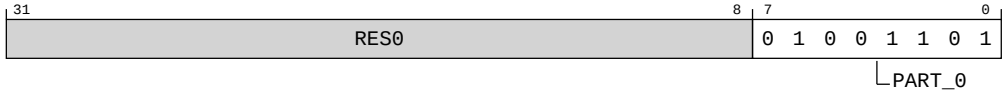


Table B-101: EDPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b01001101** A715	0x4D

Accessibility

Component	Offset	Instance	Range
Debug	0xFE0	EDPIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.14 EDPIDR1, External Debug Peripheral Identification Register 1

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* 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-62: ext_edpidr1 bit assignments

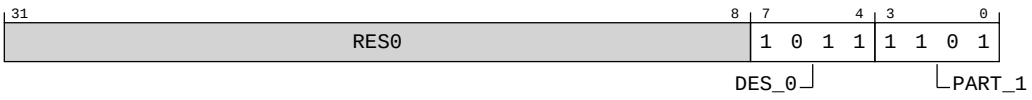


Table B-103: EDPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is `0b1011`. **`0b1011`** Arm Limited	`0b1011`
[3:0]	PART_1	Part number, most significant nibble. **`0b1101`** A715	`0b1101`

Accessibility

Component	Offset	Instance	Range
Debug	0xFE4	EDPIDR1	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.3.15 EDPIDR2, External Debug Peripheral Identification Register 2

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* 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 0001 x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-63: ext_edpidr2 bit assignments

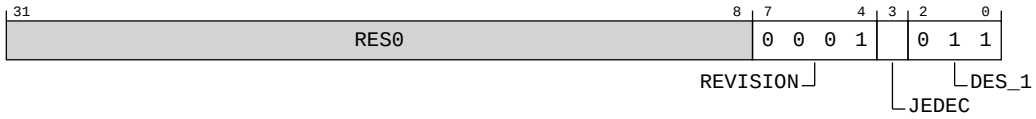


Table B-105: EDPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0001** r1p2	0b0001
[3]	JEDEC	**RAO**. Indicates a JEP106 identity code is used.	x
[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.3.16 EDPIDR3, External Debug Peripheral Identification Register 3

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* 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-64: ext_edpidr3 bit assignments

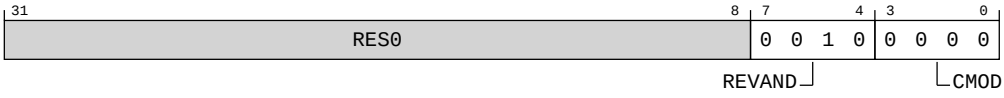


Table B-107: EDPIDR3 bit descriptions

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** r1p2	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.3.17 EDCIDR0, External Debug Component Identification Register 0

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* 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-65: ext_edcidr0 bit assignments

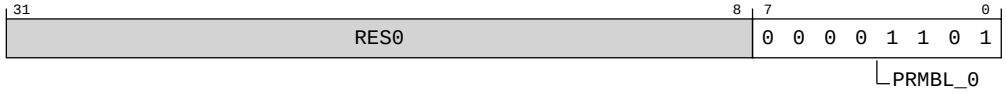


Table B-109: EDCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble.	0x0D

Accessibility

Component	Offset	Instance	Range
Debug	0xFF0	EDCIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.18 EDCIDR1, External Debug Component Identification Register 1

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* 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 1001 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-66: ext_edcidr1 bit assignments

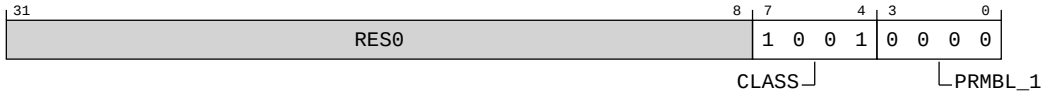


Table B-111: EDCIDR1 bit descriptions

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.	0b1001
[3:0]	PRMBL_1	Preamble.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFF4	EDCIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.19 EDCIDR2, External Debug Component Identification Register 2

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* 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-67: ext_edcidr2 bit assignments

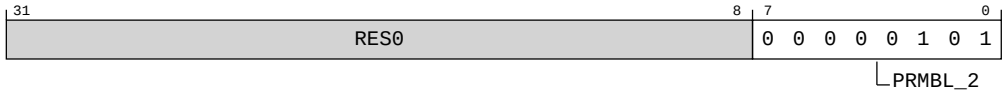


Table B-113: EDCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble.	0x05

Accessibility

Component	Offset	Instance	Range
Debug	0xFF8	EDCIDR2	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.20 EDCIDR3, External Debug Component Identification Register 3

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* 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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-68: ext_edcidr3 bit assignments

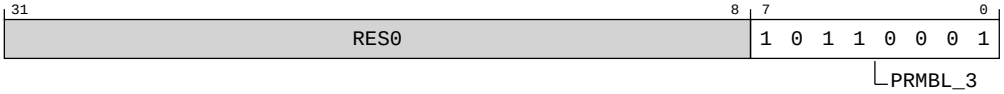


Table B-115: EDCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble.	0xB1

Accessibility

Component	Offset	Instance	Range
Debug	0xFFC	EDCIDR3	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.4 External AMU registers summary

The summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped AMU registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table B-117: AMU registers summary

Offset	Name	Reset	Width	Description
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

Offset	Name	Reset	Width	Description
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
0xCE0	AMCGCR	—	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	—	32-bit	Activity Monitors Configuration Register
0xE08	AMIIDR	—	32-bit	Activity Monitors Implementation Identification Register
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.4.1 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.9.6 AMEVTYPER00_EL0, Activity Monitors Event Type Registers 0 on page 425.

Attributes

Width

32

Component

AMU

Register offset

0x400

Access type

Read

R

Write

RESERVED

Reset value

0000 000x xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-69: ext_amevtyper00 bit assignments

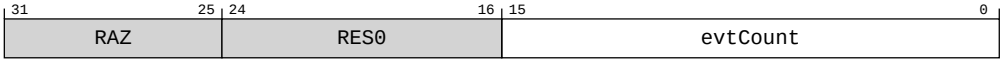


Table B-118: AMEVTYPER00 bit descriptions

Bits	Name	Description	Reset
[31:25]	**RAZ**	Reserved	**RAZ**
[24:16]	**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 required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0000000000010001** Processor frequency cycles	16{x}

B.4.2 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.9.7 AMEVTYPER01_EL0, Activity Monitors Event Type Registers 0 on page 427.

Attributes

Width

32

Component

AMU

Register offset

0x404

Access type

Read

R

Write

RESERVED

Reset value

0000 000x xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-70: ext_amevtyper01 bit assignments

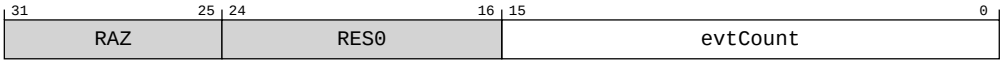


Table B-119: AMEVTYPER01 bit descriptions

Bits	Name	Description	Reset
[31:25]	**RAZ**	Reserved	**RAZ**
[24:16]	**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 required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0100000000000100** Constant frequency cycles	16{x}

B.4.3 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.9.8 AMEVTYPER02_EL0, Activity Monitors Event Type Registers 0 on page 428.

Attributes

Width

32

Component

AMU

Register offset

0x408

Access type

Read

R

Write

RESERVED

Reset value

0000 000x xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-71: ext_amevtyper02 bit assignments

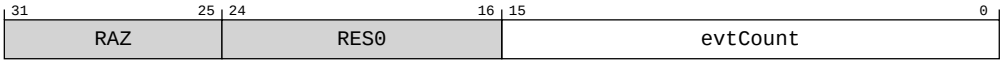


Table B-120: AMEVTYPER02 bit descriptions

Bits	Name	Description	Reset
[31:25]	**RAZ**	Reserved	**RAZ**
[24:16]	**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 required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0000000000001000** Instructions retired	`16{x}`

B.4.4 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.9.9 AMEVTYPER03_EL0, Activity Monitors Event Type Registers 0 on page 430.

Attributes

Width

32

Component

AMU

Register offset

0x40C

Access type

Read

R

Write

RESERVED

Reset value

0000 000x xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-72: ext_amevtyper03 bit assignments

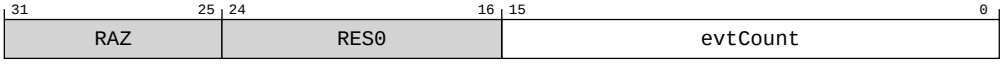


Table B-121: AMEVTYPER03 bit descriptions

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	RAZ
[24:16]	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 required for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0100000000000101** Memory stall cycles	16{x}

B.4.5 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.9.1 AMEVTYPER10_EL0, Activity Monitors Event Type Registers 1 on page 418.

Attributes

Width

32

Component

AMU

Register offset

0x480

Access type

Read

R

Write

RESERVED

Reset value

0000 000x xxxx xxxx xxxx xxxx xxxx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-73: ext_amevtyper10 bit assignments

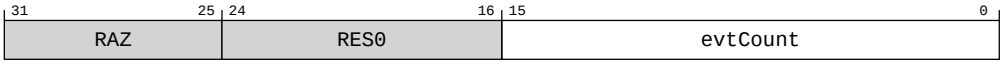


Table B-122: AMEVTYPER10 bit descriptions

Bits	Name	Description	Reset
[31:25]	**RAZ**	Reserved	**RAZ**
[24:16]	**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}`

B.4.6 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.9.2 AMEVTYPER11_EL0, Activity Monitors Event Type Registers 1 on page 419.

Attributes

Width

32

Component

AMU

Register offset

0x484

Access type

Read

R

Write

RESERVED

Reset value

0000 000x xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-74: ext_amevtyper11 bit assignments

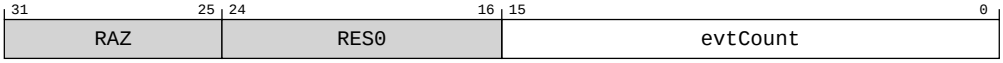


Table B-123: AMEVTYPER11 bit descriptions

Bits	Name	Description	Reset
[31:25]	**RAZ**	Reserved	**RAZ**
[24:16]	**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}`

B.4.7 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.9.3 AMEVTYPER12_EL0, Activity Monitors Event Type Registers 1 on page 421.

Attributes

Width

32

Component

AMU

Register offset

0x488

Access type

Read

R

Write

RESERVED

Reset value

0000 000x xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-75: ext_amevtyper12 bit assignments

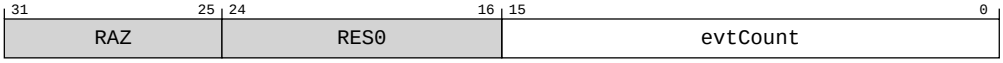


Table B-124: AMEVTYPER12 bit descriptions

Bits	Name	Description	Reset
[31:25]	**RAZ**	Reserved	**RAZ**
[24:16]	**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}`

B.4.8 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.9.5 AMCGCR_EL0, Activity Monitors Counter Group Configuration Register on page 424.

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-76: ext_amcgcr bit assignments

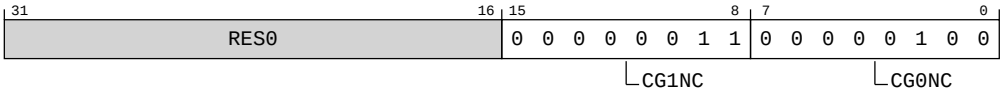


Table B-125: AMCGCR bit descriptions

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
[7:0]	CG0NC	Counter Group 0 Number of Counters. The number of counters in the architected counter group. In an implementation that includes FEAT_AMUv1, the value of this field is 4. **0b00000100** Four counters in the architected counter group	0x04

B.4.9 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.9.4 AMCFGR_EL0, Activity Monitors Configuration Register on page 422.

Attributes

Width

32

Component

AMU

Register offset

0xE00

Access type

RO

Reset value

0001 xxx1 0000 0000 0011 1111 0000 0110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-77: ext_amcfgr bit assignments

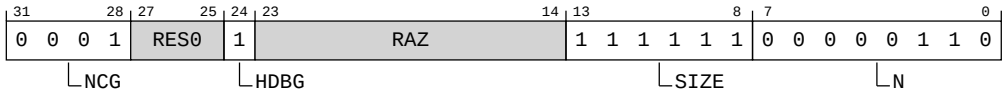


Table B-126: AMCFGR bit descriptions

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. From Armv8, this feature must be supported, and so this bit is 0b1. **0b1** ext-AMCR.HDBG is read/write.	0b1
[23:14]	**RAZ**	Reserved	**RAZ**

Bits	Name	Description	Reset
[13:8]	SIZE	Defines the size of activity monitor event counters. The size of the activity monitor event counters implemented by the Activity Monitors Extension is defined as [AMCFGR.SIZE + 1]. From Armv8, the counters are 64-bit, and so this field is `0b111111`. **Note:** Software also uses this field to determine the spacing of counters in the memory-map. From Armv8, the counters are at doubleword-aligned addresses. **`0b111111`** 64 bits	`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 defined as [AMCFGR.N + 1]. **`0b00000110`** Seven activity monitor event counters	`0x06`

B.4.10 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 0100 1101 0001 0010 0100 0011 1011

Bit descriptions

Figure B-78: ext_amiidr bit assignments

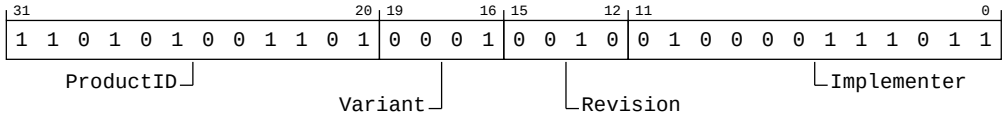


Table B-127: AMIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	This field is an AMU part identifier. **0b110101001101** A715	0xD4D
[19:16]	Variant	This field distinguishes product variants or major revisions of the product. **0b0001** r1p2	0b0001
[15:12]	Revision	This field distinguishes minor revisions of the product. **0b0010** r1p2	0b0010
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the AMU. For an Arm implementation, this field reads as 0x43B. **0b010000111011** Arm Limited	0x43B

B.4.11 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

xxxx xxxx xxxx 0000 xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-79: ext_amdevarch bit assignments

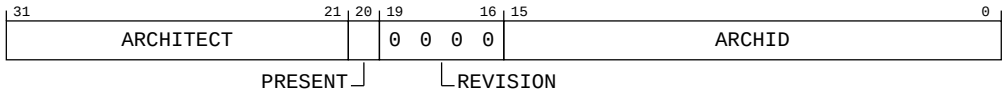


Table B-128: AMDEVARCH bit descriptions

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`.	`11{x}`
[20]	PRESENT	When set to 1, indicates that the DEVARCH is present. This field is 1 in Armv8.	`x`
[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.	`16{x}`

B.4.12 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 XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-80: ext_amdevtype bit assignments

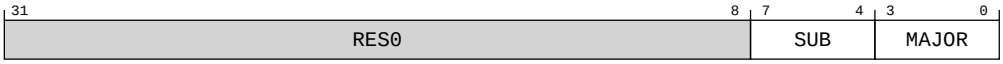


Table B-129: AMDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Subtype. Reads as `0x1`, to indicate this is a component within a PE.	xxxx
[3:0]	MAJOR	Major type. Reads as `0x6`, to indicate this is a performance monitor component.	xxxx

B.4.13 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 xxxx 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-81: ext_ampidr4 bit assignments

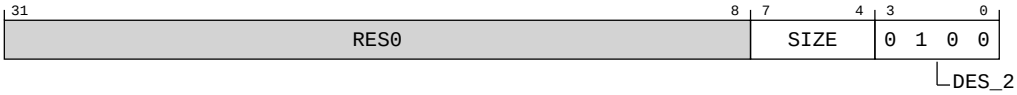


Table B-130: AMPIDR4 bit descriptions

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. This field reads as `0b0000`.	`xxxx`
[3:0]	DES_2	Designer. JEP106 continuation code, least significant nibble. For Arm Limited, this field is `0b0100`. **`0b0100`** Arm Limited	`0b0100`

B.4.14 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 0100 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-82: ext_ampidr0 bit assignments

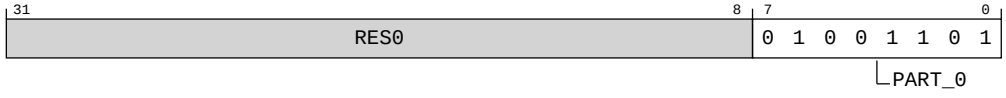


Table B-131: AMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b01001101** A715	0x4D

B.4.15 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-83: ext_ampidr1 bit assignments

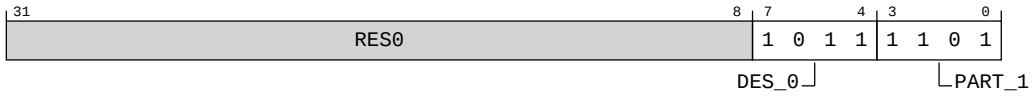


Table B-132: AMPIDR1 bit descriptions

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`

Bits	Name	Description	Reset
[3:0]	PART_1	Part number, most significant nibble. **0b1101** A715	0b1101

B.4.16 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 0001 x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-84: ext_ampidr2 bit assignments

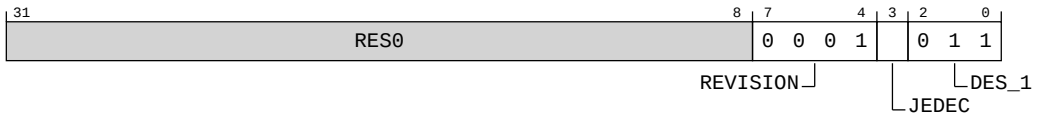


Table B-133: AMPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0001** r1p2	0b0001
[3]	JEDEC	**RAO**. Indicates a JEP106 identity code is used.	x
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. **0b011** Arm Limited	0b011

B.4.17 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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-85: ext_ampidr3 bit assignments

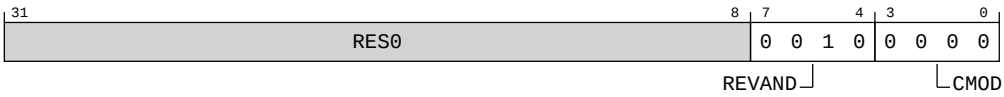


Table B-134: AMPIDR3 bit descriptions

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** r1p2	0b0010
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000**	0b0000

B.4.18 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-86: ext_amcidr0 bit assignments

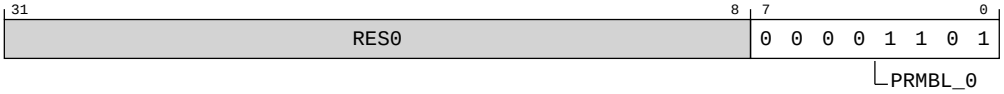


Table B-135: AMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble.	0x0D

B.4.19 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 1001 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-87: ext_amcidr1 bit assignments

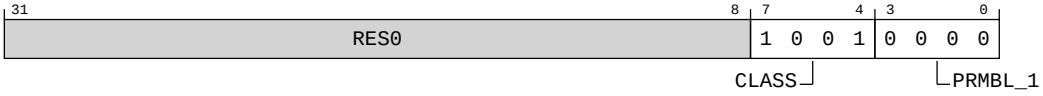


Table B-136: AMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight component. Other values are defined by the CoreSight Architecture. This field reads as 0x9.	0b1001
[3:0]	PRMBL_1	Preamble.	0b0000

B.4.20 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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-88: ext_amcidr2 bit assignments

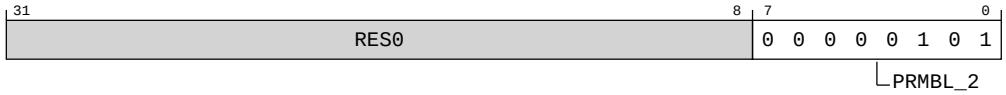


Table B-137: AMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble.	0x05

B.4.21 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-89: ext_amcidr3 bit assignments

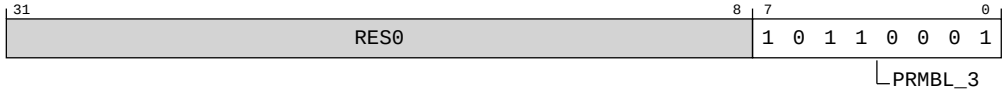


Table B-138: AMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble.	0xB1

B.5 External ETE registers summary

The summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped ETE registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table B-139: ETE registers summary

Offset	Name	Reset	Width	Description
0x018	TRCAUXCTLR	–	32-bit	Auxiliary Control Register
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

Offset	Name	Reset	Width	Description
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
0xF00	TRCITCTRL	—	32-bit	Integration Mode Control Register
0xFA0	TRCCLAIMSET	—	32-bit	Claim Tag Set Register
0xFA4	TRCCLAIMCLR	—	32-bit	Claim Tag Clear 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.5.1 TRCAUXCTLR, External 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.10.14 TRCAUXCTLR, Auxiliary Control Register on page 449.

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-90: ext_trcauxctlr bit assignments

Table B-140: 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.5.2 TRCIDR8, External 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.10.1 TRCIDR8, ID Register 8 on page 432.

Attributes

Width

32

Component

ETE

Register offset

0x180

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-91: ext_trcidr8 bit assignments

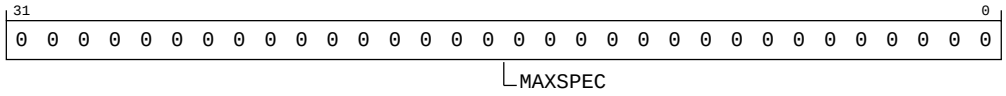


Table B-142: TRCIDR8 bit descriptions

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**	0x00000000

Accessibility

Component	Offset	Instance	Range
ETE	0x180	TRCIDR8	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.3 TRCIDR9, External 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.10.3 TRCIDR9, ID Register 9 on page 434.

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-92: ext_trcidr9 bit assignments

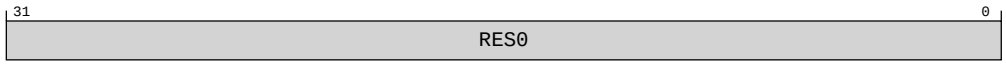


Table B-144: TRCIDR9 bit descriptions

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.5.4 TRCIDR10, External 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.10.10 TRCIDR10, ID Register 10 on page 445.

Attributes

Width

32

Component

ETE

Register offset

0x188

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-93: ext_trcidr10 bit assignments

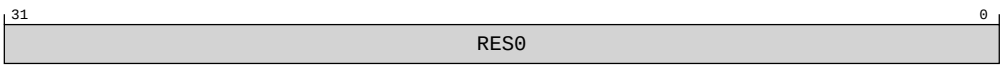


Table B-146: TRCIDR10 bit descriptions

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.5.5 TRCIDR11, External 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.10.11 TRCIDR11, ID Register 11 on page 446.

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-94: ext_trcidr11 bit assignments

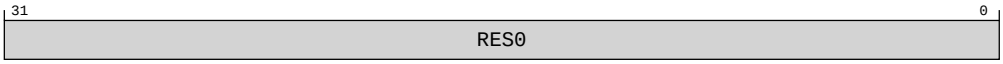


Table B-148: TRCIDR11 bit descriptions

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.5.6 TRCIDR12, External 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.10.12 TRCIDR12, ID Register 12 on page 447.

Attributes

Width

32

Component

ETE

Register offset

0x190

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-95: ext_trcidr12 bit assignments

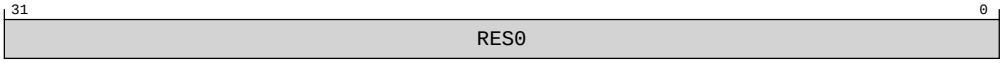


Table B-150: TRCIDR12 bit descriptions

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.5.7 TRCIDR13, External 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.10.13 TRCIDR13, ID Register 13 on page 448.

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-96: ext_trcidr13 bit assignments

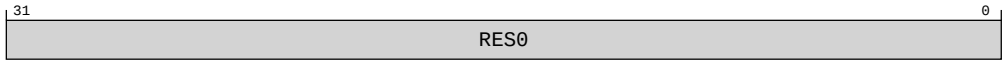


Table B-152: TRCIDR13 bit descriptions

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.5.8 TRCIMSPEC0, External 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.10.2 TRCIMSPEC0, IMP DEF Register 0 on page 433.

Attributes

Width

32

Component

ETE

Register offset

0x1C0

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-97: ext_trcimspec0 bit assignments

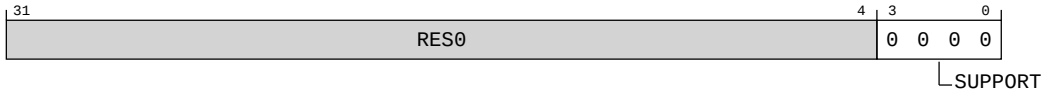


Table B-154: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[31:4]	**RES0**	Reserved	**RES0**
[3:0]	SUPPORT	Indicates whether the implementation supports **IMPLEMENTATION DEFINED** features. **0b0000** No **IMPLEMENTATION DEFINED** features are supported.	0b0000

Accessibility

Component	Offset	Instance	Range
ETE	0x1C0	TRCIMSPEC0	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.5.9 TRCIDR0, External 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.10.15 TRCIDR0, ID Register 0 on page 450.

Attributes

Width

32

Component

ETE

Register offset

0x1E0

Access type

See bit descriptions

Reset value

x010 1000 xxxx xxx0 00xx 111x 1010 000x

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-98: ext_trcidr0 bit assignments

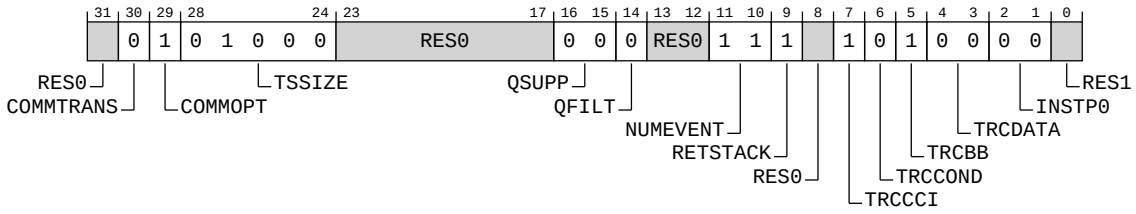


Table B-156: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[30]	COMMTRANS	Transaction Start element behavior. **0b0** Transaction Start elements are P0 elements.	0b0
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **0b1** Commit mode 1.	0b1
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. **0b01000** Global timestamping implemented with a 64-bit timestamp value.	0b01000
[23:17]	**RES0**	Reserved	**RES0**
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **0b00** Q element support is not implemented.	0b00
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	0b0
[13:12]	**RES0**	Reserved	**RES0**
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **0b11** The trace unit supports 4 ETEEvents.	0b11
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **0b1** Return stack implemented.	0b1
[8]	**RES0**	Reserved	**RES0**
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **0b1** Cycle counting implemented.	0b1
[6]	TRCCOND	Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b0** Conditional instruction tracing not implemented.	0b0
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. **0b1** Branch broadcasting implemented.	0b1
[4:3]	TRCDATA	Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Tracing of data addresses and data values is not implemented.	0b00
[2:1]	INSTP0	Indicates if load and store instructions are P0 instructions. Load and store instructions as P0 instructions is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Load and store instructions are not P0 instructions.	0b00

Bits	Name	Description	Reset
[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.5.10 TRCIDR1, External 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.10.16 TRCIDR1, ID Register 1 on page 452.

Attributes

Width

32

Component

ETE

Register offset

0x1E4

Access type

See bit descriptions

Reset value

0100 0001 xxxx xxxx xxxx 1111 1111 0001

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-99: ext_trcidr1 bit assignments

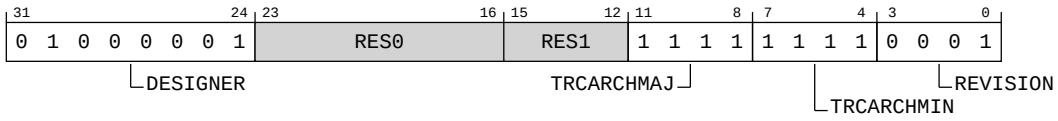


Table B-158: TRCIDR1 bit descriptions

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	0x41
[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.	0b1111
[7:4]	TRCARCHMIN	Minor architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	0b1111
[3:0]	REVISION	Indicates the major revision of the product **0b0001** r1p2	0b0001

Accessibility

Component	Offset	Instance	Range
ETE	0x1E4	TRCIDR1	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.11 TRCIDR2, External 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.10.4 TRCIDR2, ID Register 2 on page 435.

Attributes

Width

32

Component

ETE

Register offset

0x1E8

Access type

See bit descriptions

Reset value

1100 000x xxxx xxxx x001 0000 1000 1000

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-100: ext_trcidr2 bit assignments

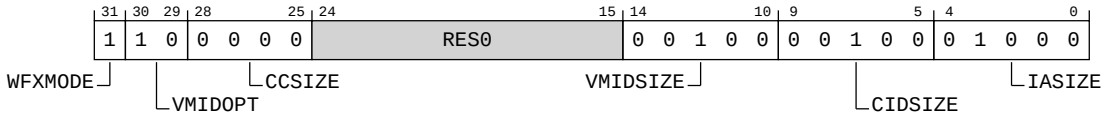


Table B-160: TRCIDR2 bit descriptions

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.	0b1
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. **0b10** Virtual context identifier selection not supported. ext-TRCCONFIGR.VMIDOPT is **RES1**.	0b10

Bits	Name	Description	Reset
[28:25]	CCSIZE	Indicates the size of the cycle counter. **0b0000** The cycle counter is 12 bits in length.	0b0000
[24:15]	**RES0**	Reserved	**RES0**
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. **0b00100** 32-bit Virtual context identifier size.	0b00100
[9:5]	CIDSIZE	Indicates the Context identifier size. **0b00100** 32-bit Context identifier size.	0b00100
[4:0]	IASIZE	Virtual instruction address size. **0b01000** Maximum of 64-bit instruction address size.	0b01000

Accessibility

Component	Offset	Instance	Range
ETE	0x1E8	TRCIDR2	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RO

B.5.12 TRCIDR3, External 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.10.5 TRCIDR3, ID Register 3 on page 437.

Attributes

Width
32

Component
ETE

Register offset
0x1EC

Access type

See bit descriptions

Reset value

0000 0001 x111 1111 xx00 0000 0000 0100

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-101: ext_trcidr3 bit assignments

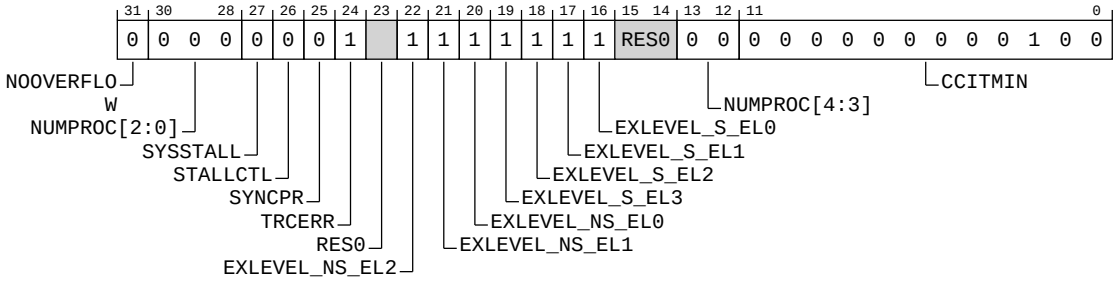


Table B-162: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	0b0
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **0b0** Stalling of the PE is not permitted.	0b0
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **0b0** Stalling of the PE is not implemented.	0b0
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0b0** ext-TRCSYNCPR is read/write so software can change the synchronization period.	0b0
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **0b1** Forced tracing of System Error exceptions is implemented.	0b1
[23]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 is implemented. **0b1** Non-secure EL2 is implemented.	0b1
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 is implemented. **0b1** Non-secure EL1 is implemented.	0b1
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 is implemented. **0b1** Non-secure EL0 is implemented.	0b1
[19]	EXLEVEL_S_EL3	Indicates if EL3 is implemented. **0b1** EL3 is implemented.	0b1
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 is implemented. **0b1** Secure EL2 is implemented.	0b1
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 is implemented. **0b1** Secure EL1 is implemented.	0b1
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 is implemented. **0b1** Secure EL0 is implemented.	0b1
[15:14]	**RES0**	Reserved	**RES0**
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. **0b00000** The trace unit can trace one PE.	0b00000
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in 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**	0x004

Accessibility

Component	Offset	Instance	Range
ETE	0x1EC	TRCIDR3	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.13 TRCIDR4, External 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.10.6 TRCIDR4, ID Register 4 on page 439.

Attributes

Width

32

Component

ETE

Register offset

0x1F0

Access type

See bit descriptions

Reset value

0001 0001 0001 0111 0000 xxx0 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-102: ext_trcidr4 bit assignments

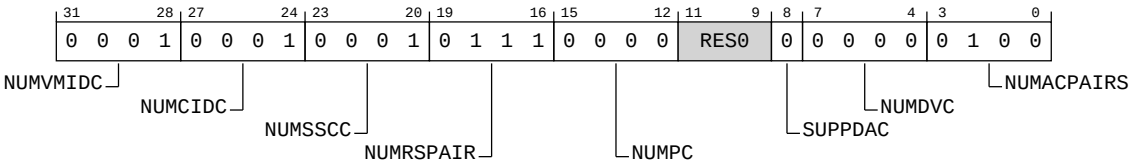


Table B-164: TRCIDR4 bit descriptions

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.	0b0001

Bits	Name	Description	Reset
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Context Identifier Comparator.	0b0001
[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.	0b0001
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0b0111** The implementation has eight resource selector pairs.	0b0111
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0b0000** No PE Comparator Inputs are available.	0b0000
[11:9]	**RES0**	Reserved	**RES0**
[8]	SUPPDAC	Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0** Data address comparisons not implemented.	0b0
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0000** No data value comparators implemented.	0b0000
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **0b0100** The implementation has four Address Comparator pairs.	0b0100

Accessibility

Component	Offset	Instance	Range
ETE	0x1F0	TRCIDR4	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.14 TRCIDR5, External 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.10.7 TRCIDR5, ID Register 5 on page 441.

Attributes

Width

32

Component

ETE

Register offset

0x1F4

Access type

See bit descriptions

Reset value

x010 100x 0100 0111 xxxx 1001 1111 1111

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-103: ext_trcidr5 bit assignments

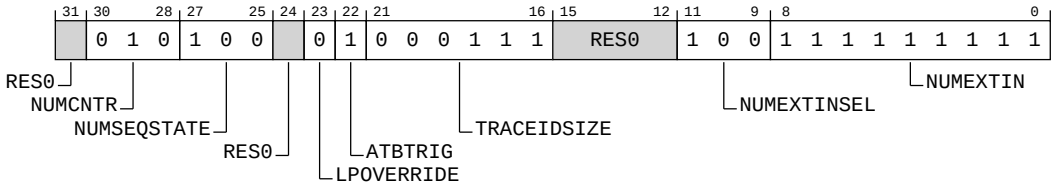


Table B-166: TRCIDR5 bit descriptions

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.	0b010

Bits	Name	Description	Reset
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **0b100** Four Sequencer states are implemented.	0b100
[24]	**RES0**	Reserved	**RES0**
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **0b0** The trace unit does not support Low-power Override Mode.	0b0
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **0b1** The implementation supports ATB triggers.	0b1
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **0b000111** The implementation supports a 7-bit trace ID.	0b000111
[15:12]	**RES0**	Reserved	**RES0**
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. **0b100** 4 External Input Selector resources are available.	0b100
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **0b111111111** Unified PMU event selection.	0b111111111

Accessibility

Component	Offset	Instance	Range
ETE	0x1F4	TRCIDR5	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RO

B.5.15 TRCIDR6, External 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.10.8 TRCIDR6, ID Register 6 on page 443.

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-104: ext_trcidr6 bit assignments

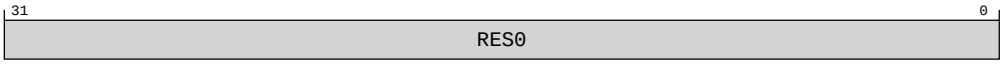


Table B-168: TRCIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x1F8	TRCIDR6	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.16 TRCIDR7, External 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.10.9 TRCIDR7, ID Register 7 on page 444.

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-105: ext_trcidr7 bit assignments

Table B-170: 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.5.17 TRCITCTRL, Integration Mode Control Register

A component can use TRCITCTRL to dynamically switch between functional mode and integration mode. In integration mode, topology detection is enabled. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xF00

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-106: ext_trcitctrl bit assignments

Table B-172: 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.5.18 TRCCLAIMSET, External 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.10.19 TRCCLAIMSET, Claim Tag Set Register on page 456.

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-107: ext_trcclaimset bit assignments

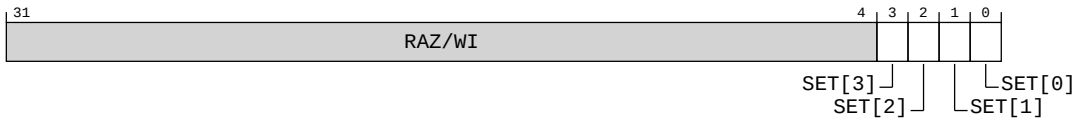


Table B-174: TRCCLAIMSET bit descriptions

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
[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

Accessibility

Component	Offset	Instance	Range
ETE	0xFA0	TRCCLAIMSET	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RW

B.5.19 TRCCLAIMCLR, External 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.10.20 TRCCLAIMCLR, Claim Tag Clear Register on page 459.

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-108: ext_trcclaimclr bit assignments

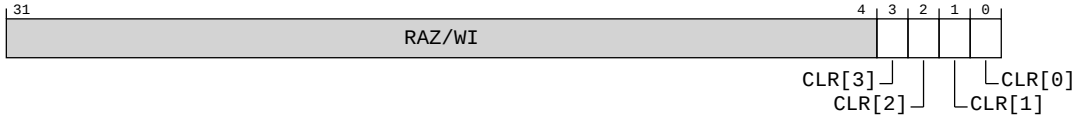


Table B-176: TRCCLAIMCLR bit descriptions

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
[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

Bits	Name	Description	Reset
[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.5.20 TRCDEVARCH, External 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.10.17 TRCDEVARCH, Device Architecture Register on page 453.

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-109: ext_trcdevarch bit assignments

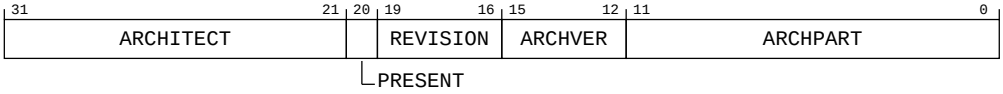


Table B-178: TRCDEVARCH bit descriptions

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. **0b0000** ETEv1.0, FEAT_ETE. All other values are reserved.	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}

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFBC	TRCDEVARCH	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.21 TRCDEVID2, 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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-110: ext_trcdevid2 bit assignments

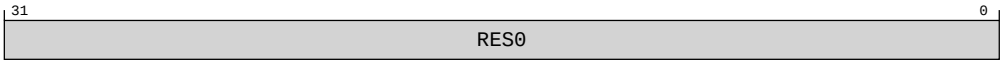


Table B-180: TRCDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFC0	TRCDEVID2	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.22 TRCDEVID1, 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-111: ext_trcdevid1 bit assignments

Table B-182: TRCDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFC4	TRCDEVID1	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.23 TRCDEVID, External 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.10.18 TRCDEVID, Device Configuration Register on page 455.

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-112: ext_trcdevid bit assignments

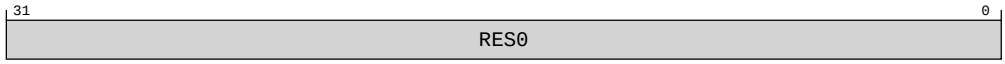


Table B-184: TRCDEVID bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFC8	TRCDEVID	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.24 TRCDEVTYPE, 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 0001 0011

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-113: ext_trcdevtype bit assignments

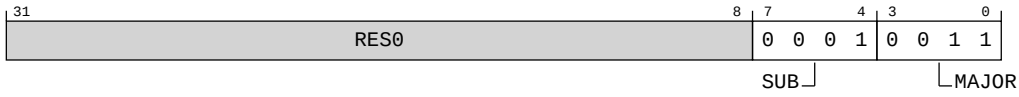


Table B-186: TRCDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7:4]	SUB	Component sub-type. **0b0001** When MAJOR == 0x3 (Trace source): Associated with a PE. This field reads as 0x1.	0b0001
[3:0]	MAJOR	Component major type. **0b0011** Trace source. Other values are defined by the CoreSight Architecture. This field reads as 0x3.	0b0011

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFCC	TRCDEVTYPE	None

This interface is accessible as follows:

When !IsTraceCorePowered()
ERROR

Otherwise
RO

B.5.25 TRCPIDR4, 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-114: ext_trcpidr4 bit assignments

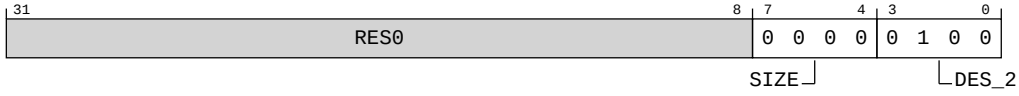


Table B-188: TRCPIDR4 bit descriptions

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. 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.5.26 TRCPIDR5, Peripheral Identification Register 5

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFD4

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-115: ext_trcpidr5 bit assignments

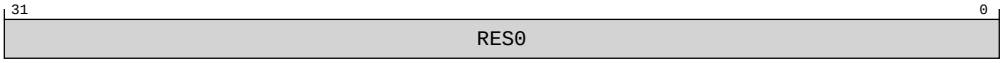


Table B-190: TRCPIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFD4	TRCPIDR5	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.27 TRCPIDR6, 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-116: ext_trcpidr6 bit assignments

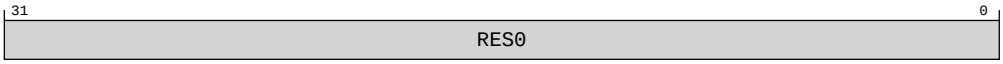


Table B-192: TRCPIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFD8	TRCPIDR6	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.28 TRCPIDR7, 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-117: ext_trcpidr7 bit assignments

Table B-194: TRCPIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFDC	TRCPIDR7	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.29 TRCPIDR0, 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 0100 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-118: ext_trcpidr0 bit assignments

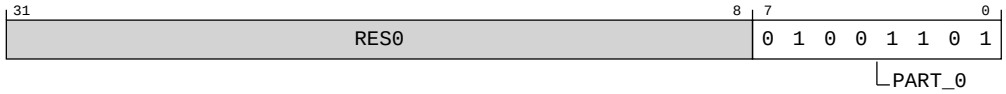


Table B-196: TRCPIDR0 bit descriptions

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. **0b01001101** A715	0x4D

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFE0	TRCPIDR0	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.30 TRCPIDR1, 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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-119: ext_trcpidr1 bit assignments

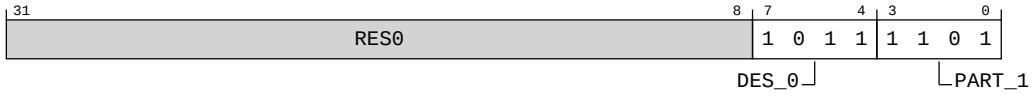


Table B-198: TRCPIDR1 bit descriptions

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** A715	0b1101

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFE4	TRCPIDR1	None

This interface is accessible as follows:

When !IsTraceCorePowered()
ERROR

Otherwise
RO

B.5.31 TRCPIDR2, 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 0001 1011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-120: ext_trcpidr2 bit assignments

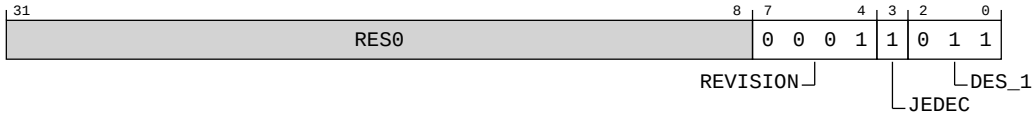


Table B-200: TRCPIDR2 bit descriptions

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. **`0b0001`** r1p2	`0b0001`
[3]	JEDEC	JEDEC-assigned JEP106 implementer code is used.	`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.5.32 TRCPIDR3, Peripheral Identification Register 3

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFEC

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-121: ext_trcpidr3 bit assignments

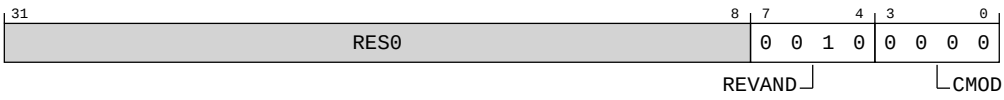


Table B-202: TRCPIDR3 bit descriptions

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`** r1p2	`0b0010`
[3:0]	CMOD	Customer Modified. Indicates the component has been modified. A value of `0b0000` means the component is not modified from the original design. Any other value means the component has been modified in an **IMPLEMENTATION DEFINED** way. **`0b0000`**	`0b0000`

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFEC	TRCPIDR3	None

This interface is accessible as follows:

When !IsTraceCorePowered()
ERROR

Otherwise
RO

B.5.33 TRCCIDR0, 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-122: ext_trccidr0 bit assignments

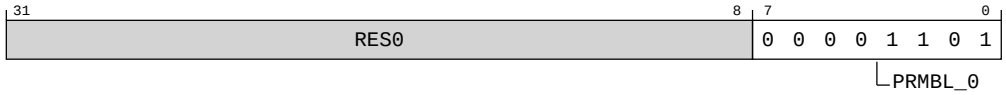


Table B-204: TRCCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Component identification preamble, segment 0.	0x0D

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFF0	TRCCIDR0	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.34 TRCCIDR1, 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 1001 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-123: ext_trccidr1 bit assignments

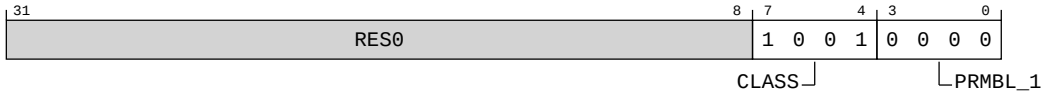


Table B-206: TRCCIDR1 bit descriptions

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.	0b1001
[3:0]	PRMBL_1	Component identification preamble, segment 1.	0b0000

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFF4	TRCCIDR1	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.35 TRCCIDR2, 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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-124: ext_trccidr2 bit assignments

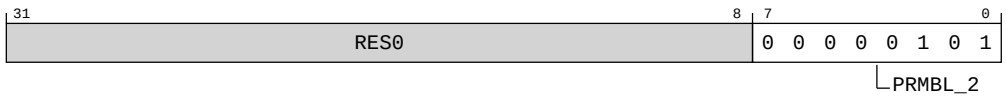


Table B-208: TRCCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	PRMBL_2	Component identification preamble, segment 2.	0x05

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFF8	TRCCIDR2	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.5.36 TRCCIDR3, 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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-125: ext_trccidr3 bit assignments

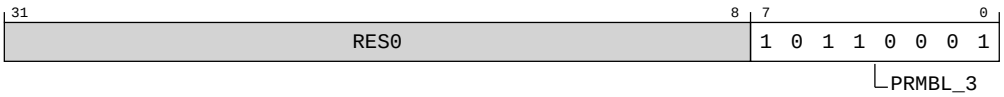


Table B-210: TRCCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Component identification preamble, segment 3.	0xB1

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFFC	TRCCIDR3	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.6 External ROM table registers summary

The summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped ROM table registers in the core. For more information about a register, click the register name in the table.

For registers without a listed reset value refer to the individual field resets documented on the register description pages or in the Arm ARM.

Table B-212: 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
0xFB8	AUTHSTATUS	–	32-bit	Authentication Status Register
0xFBC	DEVARCH	–	32-bit	Device Architecture Register
0xFD0	PIDR4	–	32-bit	Peripheral Identification Register 4
0xFE0	PIDR0	–	32-bit	Peripheral Identification Register 0

Offset	Name	Reset	Width	Description
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.6.1 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 0, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class 0x9 ROM Table:

- The first entry, entry 0, has offset 0x000.
- ext-ROMENTRY<n> has the offset 0x000 + 0×4, where 0 ≤ 0 ≤ 511.
- If the number of components, 0, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:
  - ROM Table entries representing components have offsets from 0x000 to (0-1)×4.
  - The ext-ROMENTRY<n> at offset 0×4, which has a PRESENT field with the value 0b00, indicates the end of the ROM Table.
- If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from 0x000 to 0x7FC. If a ROM Table entry is present at offset 0x7FC, its PRESENT field must have a value of either 0b00 or 0b11, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value 0b11.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

If ext-DEVID.FORMAT has the value 0x0, ext-ROMENTRY<n> are 512 32-bit registers.

If ext-DEVID.FORMAT has the value 0x1, ext-ROMENTRY<n> are 256 64-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0x0

Access type

Read

R

Write

RESERVED

Reset value

0000 0000 0000 0001 0000 xxx0 0000 x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-126: ext_romentry0 bit assignments

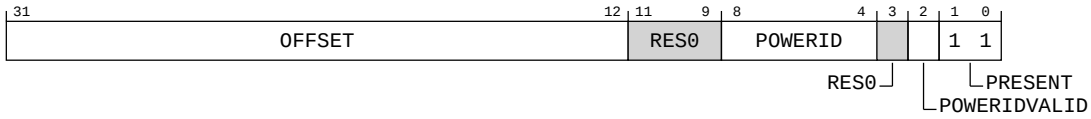


Table B-213: ROMENTRY0 bit descriptions

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 Debug	0x00010
[11:9]	**RES0**	Reserved	**RES0**
[8:4]	POWERID	The power domain ID of the component. This field supports up to 32 power domains using values 0x00 to 0x1F.	0b00000
[3]	**RES0**	Reserved	**RES0**
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0b0** A power domain ID is not provided.	0b0

Bits	Name	Description	Reset
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **0b11** The ROM Entry is present.	0b11

B.6.2 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 1, 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 + 1×4, where 0 ≤ 1 ≤ 511.
- If the number of components, 1, 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 (1-1)×4.
  - The ext-ROMENTRY<n> at offset 1×4, 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.

If ext-DEVID.FORMAT has the value 0x1, ext-ROMENTRY<n> are 256 64-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0x4

Access type

Read

R

Write

RESERVED

Reset value

0000 0000 0000 0010 0000 xxx0 0000 x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-127: ext_romentry1 bit assignments

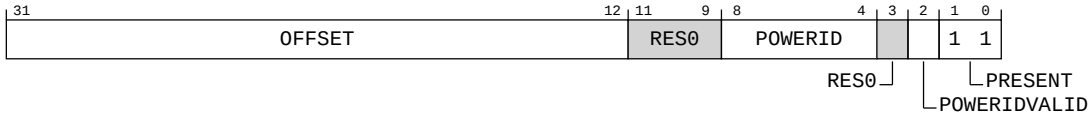


Table B-214: ROMENTRY1 bit descriptions

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 PMU	0x00020
[11:9]	**RES0**	Reserved	**RES0**
[8:4]	POWERID	The power domain ID of the component. This field supports up to 32 power domains using values 0x00 to 0x1F.	0b00000
[3]	**RES0**	Reserved	**RES0**
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0b0** A power domain ID is not provided.	0b0

Bits	Name	Description	Reset
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **0b11** The ROM Entry is present.	0b11

B.6.3 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 2, 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 + 2×4, where 0 ≤ 2 ≤ 511.
- If the number of components, 2, 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 (2-1)×4.
  - The ext-ROMENTRY<n> at offset 2×4, 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.

If ext-DEVID.FORMAT has the value 0x1, ext-ROMENTRY<n> are 256 64-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0x8

Access type

Read

R

Write

RESERVED

Reset value

0000 0000 0000 0011 0000 xxx0 0000 x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-128: ext_romentry2 bit assignments

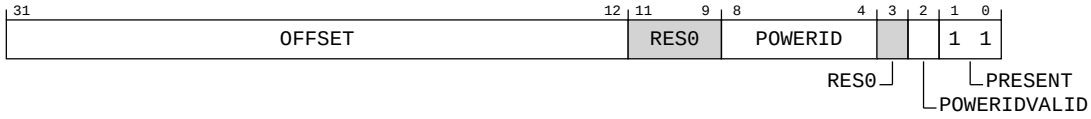


Table B-215: ROMENTRY2 bit descriptions

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 ETM	0x00030
[11:9]	**RES0**	Reserved	**RES0**
[8:4]	POWERID	The power domain ID of the component. This field supports up to 32 power domains using values 0x00 to 0x1F.	0b00000
[3]	**RES0**	Reserved	**RES0**
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0b0** A power domain ID is not provided.	0b0

Bits	Name	Description	Reset
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **0b11** The ROM Entry is present.	0b11

B.6.4 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 3, 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 + 3×4, where 0 ≤ 3 ≤ 511.
- If the number of components, 3, 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 (3-1)×4.
  - The ext-ROMENTRY<n> at offset 3×4, 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.

If ext-DEVID.FORMAT has the value 0x1, ext-ROMENTRY<n> are 256 64-bit registers.

Attributes

Width

32

Component

ROM table

Register offset

0xC

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxx0 0000 x0xx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-129: ext_romentry3 bit assignments

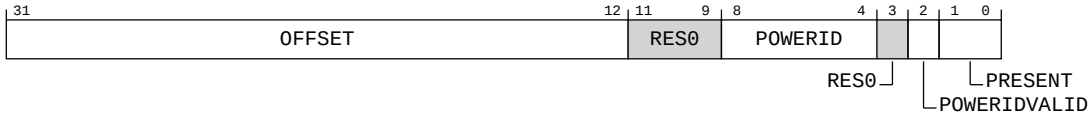


Table B-216: ROMENTRY3 bit descriptions

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. **0b00000000000000000000** ROM Entry is not present. **0b00000000000001000000** Core ELA	20{x}
[11:9]	**RES0**	Reserved	**RES0**
[8:4]	POWERID	The power domain ID of the component. This field supports up to 32 power domains using values 0x00 to 0x1F.	0b00000
[3]	**RES0**	Reserved	**RES0**
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0b0** A power domain ID is not provided.	0b0

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, and this ext-ROMENTRY3 is the final entry in the ROM Table. If PRESENT has this value, all other fields in this ext-ROMENTRY3 must be zero. **0b11** The ROM Entry is present.	The reset values can be the following: 0b00, 0b11, respective to the value.

B.6.5 AUTHSTATUS, Authentication Status Register

AUTHSTATUS indicates whether certain functions are enabled.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ROM table

Register offset

0xFB8

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-130: ext_authstatus bit assignments

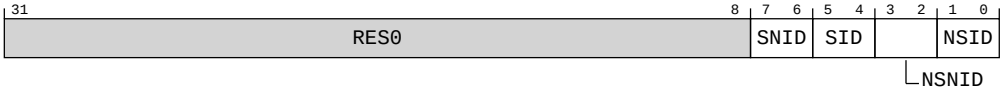


Table B-217: AUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:6]	SNID	Secure Non-invasive Debug. **0b00** Debug level is not supported. **0b10** Supported and disabled. (SPIDEN | SPNIDEN) & (DBGEN | NIDEN) == FALSE. **0b11** Supported and enabled. (SPIDEN | SPNIDEN) & (DBGEN | NIDEN) == TRUE.	xx
[5:4]	SID	Secure Invasive Debug. **0b00** Debug level is not supported. **0b10** Secure invasive debug disabled. (SPIDEN & DBGEN) == FALSE. **0b11** Secure invasive debug enabled. (SPIDEN & DBGEN) == TRUE.	xx
[3:2]	NSNID	Non-secure Non-invasive Debug. **0b00** Debug level is not supported. **0b10** Supported and disabled. (NIDEN | DBGEN) == FALSE. **0b11** Supported and enabled. (NIDEN | DBGEN) == TRUE.	xx
[1:0]	NSID	Non-secure Invasive Debug. **0b00** Debug level is not supported. **0b10** Supported and disabled. DBGEN == FALSE. **0b11** Supported and enabled. DBGEN == TRUE.	xx

B.6.6 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

0100 0111 0111 0000 0000 1010 1111 0111

Bit descriptions

Figure B-131: ext_devarch bit assignments

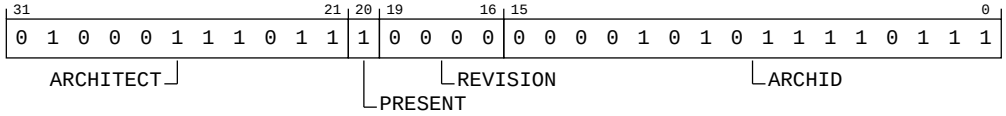


Table B-218: DEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. **0b01000111011** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	0b01000111011
[20]	PRESENT	Present. **0b1** DEVARCH information present.	0b1
[19:16]	REVISION	Revision. **0b0000** Revision 0.	0b0000
[15:0]	ARCHID	Architecture ID. **0b0000101011110111** ROM Table v0. The debug tool must inspect ext-DEVTYPE and ext-DEVID to determine further information about the ROM Table.	0x0AF7

B.6.7 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 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-132: ext_pidr4 bit assignments

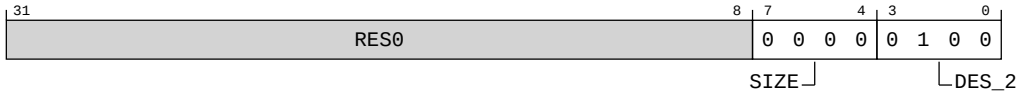


Table B-219: PIDR4 bit descriptions

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.	0b0000
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **0b0100** Arm Limited	0b0100

B.6.8 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 0100 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-133: ext_pidr0 bit assignments

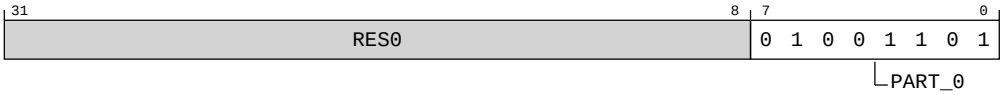


Table B-220: PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b01001101** A715	0x4D

B.6.9 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 1011 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-134: ext_pidr1 bit assignments

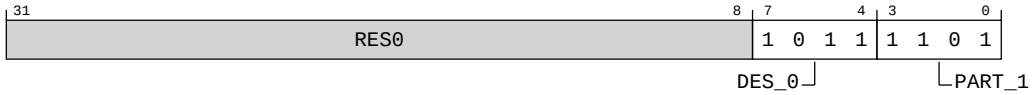


Table B-221: PIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is `0b1011`. **`0b1011`** Arm Limited	`0b1011`
[3:0]	PART_1	Part number, most significant nibble. **`0b1101`** A715	`0b1101`

B.6.10 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 0001 x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-135: ext_pidr2 bit assignments

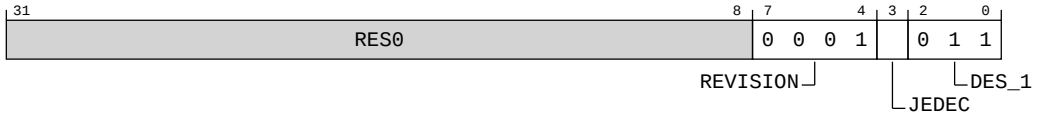


Table B-222: PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0001** r1p2	0b0001
[3]	JEDEC	**RAO**. Indicates a JEP106 identity code is used.	x
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. **0b011** Arm Limited	0b011

B.6.11 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 0010 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-136: ext_pidr3 bit assignments

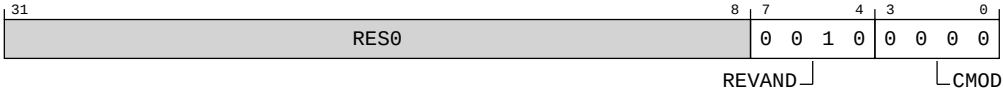


Table B-223: PIDR3 bit descriptions

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** r1p2	0b0010
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000**	0b0000

B.6.12 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 0000 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-137: ext_cidr0 bit assignments

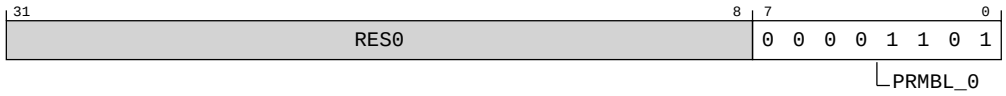


Table B-224: CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	CoreSight component identification preamble. **0b00001101** CoreSight component identification preamble.	0x0D

B.6.13 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 1001 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-138: ext_cidr1 bit assignments

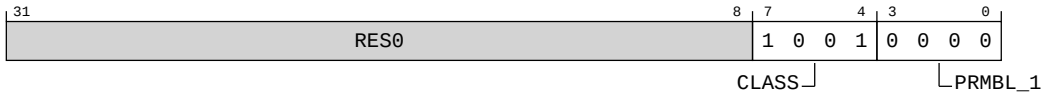


Table B-225: CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	CoreSight component class. **0b1001** CoreSight component.	0b1001
[3:0]	PRMBL_1	CoreSight component identification preamble. **0b0000** CoreSight component identification preamble.	0b0000

B.6.14 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 0000 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-139: ext_cidr2 bit assignments

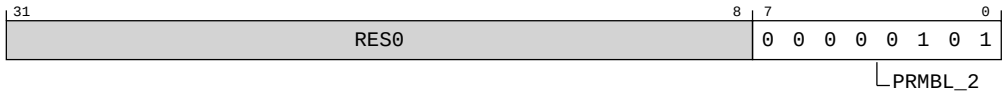


Table B-226: CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	CoreSight component identification preamble. **0b00000101** CoreSight component identification preamble.	0x05

B.6.15 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 1011 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-140: ext_cidr3 bit assignments

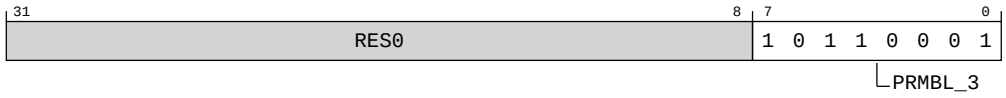


Table B-227: CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	CoreSight component identification preamble. **0b10110001** CoreSight component identification preamble.	0xB1

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 0000-02

Change	Location
First Confidential alpha release for r0p0	-

Table C-2: Differences between issue 0000-02 and issue 0000-03

Change	Location
First Confidential beta release for r0p0	-
Added a reminder that this document does not include the complete list of registers	2. The Cortex-A715 core on page 22
Renamed MMU Translation Cache to L2 TLB	2.1 Cortex-A715 core features on page 22
Added a new topic on DSU-110 dependent features	2.3 DSU-110 dependent features on page 25
Added 'Enhanced PAN' and 'MTE Asymmetric Fault Handling' optional features	2.4 Supported standards and specifications on page 25
Added sections on SPE and AMU	3.1 Core components on page 31
Added links to TRBE, AMU, and SPE in the 'Related information' section	3.1 Core components on page 31
Renamed MMU Translation Cache to L2 TLB	6.4 Translation table walks on page 54
Added explanation of GRE	6.7 Memory behavior and supported memory types on page 57
Updated registers with the ERR0 or ERRx prefix to ERR1 within the chapter	11. RAS Extension support on page 77
Updated register ERR0PFGCDN to ERR1PFGCDN and changed referenced manual for more information	11.5 Error injection on page 80
Added BFloat16 and Int8 Matrix Multiplication support	14. Advanced SIMD and floating-point support on page 86
Removed AArch64 Debug register summary section within the chapter	17. Debug on page 89
Added section	17.6 CoreSight component identification on page 94
Updated CommonEvent PMU events table	18.1 Performance monitors events on page 98
Minor changes	22. Statistical Profiling Extension Support on page 127
Added a reminder that this document does not include the complete list of registers	A. AArch64 registers on page 130
Updated the description of some registers	A. AArch64 registers on page 130
Removed Debug register	A. AArch64 registers on page 130

Change	Location
Added a reminder that this document does not include the complete list of registers	B. External registers on page 468
Updated the description of some registers	B. External registers on page 468

Table C-3: Differences between issue 0000-03 and issue 0000-04

Change	Location
First Confidential limited access release for r0p0	-
Updated the pdf style including wider tables, notes presentation, tables numbering, figures numbering	Throughout document
Editorial changes	Throughout document
Removed a note and placed it in the dedicated cache memory chapters	2.2 Cortex-A715 core configuration options on page 24
Editorial changes to clarify the information (mainly adding some of the features name using 'FEAT_xxx' terminology)	2.4 Supported standards and specifications on page 25
Armv8.2-VPIPT, VMID-aware PIPT instruction cache feature not supported and removed from the list of optional features	2.4 Supported standards and specifications on page 25
Updated core components figure adding 'Dynamic branch predictor' and 'AMU' blocs	3.1 Core components on page 31
Added SHA-3 to the list of SHA functions supported by the Cryptographic Extension	3.1 Core components on page 31
Added SVE2 to the Scalable Vector Extension description	3.1 Core components on page 31
Removed EVENTIREQ signal from WFI state exit conditions	5.2.1 Wait for Interrupt and Wait for Event on page 40
Removed information on PDCORE0 and PDCORE1 that are not relevant	5.3 Power control on page 41
Updated the Caution note using the information previously located after the table	5.4 Core power modes on page 42
Editorial changes: added a link to registers	5.4.4 Full retention mode on page 44
Added chapter	5.5 Performance and power management on page 46
Added related information	5.6 Cortex-A715 core powerup and powerdown sequence on page 48
Added the values taken by 'x' to clarify TCR_Elx meaning	6.5 Hardware management of the Access flag and dirty state on page 55
Editorial changes to clarify the information on Misprogramming continuous hints	6.6 Responses on page 55
Added a note	8.1 L1 data cache behavior on page 63
Added section	8.2 Write streaming mode on page 64
Added L2 to the list of targeted caches	8.5 Data prefetching on page 66
Added a note	9.1 L2 cache on page 68
Added information	9.2 Support for memory types on page 69
Added chapter	10. Direct access to internal memory on page 71
Updated initial description	11.3 Fault detection and reporting on page 79
Updated initial description	13.1 Disable the GIC CPU interface on page 84
Added explanations	17. Debug on page 89
Added explanations	17.2.4 Breakpoints and watchpoints on page 93
Removed 'ETM exported' column	18.1 Performance monitors events on page 98
Removed duplicated information in the table	18.1 Performance monitors events on page 98

Change	Location
Removed events 0x001A and 0x0072 that are not supported anymore	18.1 Performance monitors events on page 98
Added events 0x818, 0x8129, 0x8136, 0x8137, 0x8162	18.1 Performance monitors events on page 98
Updated counters reference for Group 1 (10-12 instead of 0-2)	21. Activity Monitors Extension support on page 123
Updated the description of some registers	A. AArch64 registers on page 130
Removed AArch64 Statistical Profiling Extension registers	A. AArch64 registers on page 130
Updated accessibility code	A.1.1 AIDR_EL1, Auxiliary ID Register on page 131
Updated AMAIR_EL1 accessibility code	A.1.6 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2) on page 141
Updated accessibility code	A.1.7 LORID_EL1, LORegionID (EL1) on page 144
Updated AMAIR_EL1 accessibility code	A.1.8 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register (EL1) on page 145
Updated accessibility code	A.1.11 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register on page 151
Updated accessibility code	A.1.12 IMP_CPUACTLR2_EL1, CPU Auxiliary Control Register on page 153
Updated accessibility code	A.1.13 IMP_CPUACTLR3_EL1, CPU Auxiliary Control Register on page 155
Updated accessibility code	A.1.14 IMP_CPUACTLR4_EL1, CPU Auxiliary Control Register on page 157
Added bits in the register description, and updated accessibility code	A.1.15 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 159
Added bits in the register description, and updated accessibility code	A.1.16 IMP_CPUECTLR2_EL1, CPU Extended Control Register on page 167
Removed VPU_PWR_CTRL bit that is not supported, and updated accessibility code	A.1.17 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 170
Updated accessibility code	A.1.18 IMP_CLUSTERACTLR_EL1, Cluster Auxiliary Control Register on page 173
IMP_ATCR_EL2 register added to AArch64 Generic system control registers	A.1.19 IMP_ATCR_EL2, CPU Auxiliary Translation Control Register on page 175
Added missing word (virtualization) to register name, and changed 'page table walks' to 'translation table walks'	A.1.20 IMP_AVTCR_EL2, CPU Auxiliary Virtualization Translation Control Register on page 179
Changed 'page table walks' to 'translation table walks'	A.1.21 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register on page 182
Updated bits description	A.1.22 IMP_ISIDE_DATA0_EL3, RAMINDEX Instruction Data register 0 on page 185
Updated bits description	A.1.23 IMP_ISIDE_DATA1_EL3, RAMINDEX Instruction Data register 1 on page 187
Updated bits description	A.1.24 IMP_ISIDE_DATA2_EL3, RAMINDEX Instruction Data register 2 on page 189
Updated bits description	A.1.25 IMP_MMU_DATA0_EL3, RAMINDEX TLB Data register 0 on page 191
Updated bits description	A.1.26 IMP_MMU_DATA1_EL3, RAMINDEX TLB Data register 1 on page 195
Updated bits description	A.1.27 IMP_MMU_DATA2_EL3, RAMINDEX TLB Data register 2 on page 201

Change	Location
Updated bits description	A.1.28 IMP_DSIDE_DATA0_EL3, RAMINDEX L1D Data register 0 on page 206
Updated bits description	A.1.29 IMP_DSIDE_DATA1_EL3, RAMINDEX L1D Data register 1 on page 208
Updated bits description	A.1.30 IMP_DSIDE_DATA2_EL3, RAMINDEX L1D Data register 2 on page 210
IMP_L2_DATA0_EL3 register added to AArch64 Generic system control registers	A.1.31 IMP_L2_DATA0_EL3, RAMINDEX L2 Data register 0 on page 212
IMP_L2_DATA2_EL3 register added to AArch64 Generic system control registers	A.1.32 IMP_L2_DATA2_EL3, RAMINDEX L2 Data register 2 on page 216
IMP_L2_DATA1_EL3 register added to AArch64 Generic system control registers	A.1.33 IMP_L2_DATA1_EL3, RAMINDEX L2 Data register 1 on page 218
Changed 'page table walks' to 'translation table walks'	A.1.35 IMP_ATCR_EL3, CPU Auxiliary Translation Control Register on page 222
Added FPCR register to the AArch64 generic system control registers	A.1.43 FPCR, Floating-point Control Register on page 235
Updated AFSR0_EL1 accessibility code	A.1.44 AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2) on page 239
Updated AFSR1_EL1 accessibility code	A.1.45 AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2) on page 242
Updated AFSR0_EL1 accessibility code	A.1.46 AFSR0_EL1, Auxiliary Fault Status Register 0 (EL1) on page 244
Updated AFSR1_EL1 accessibility code	A.1.47 AFSR1_EL1, Auxiliary Fault Status Register 1 (EL1) on page 247
Added AArch64 system instruction registers	A.3 AArch64 This register is a system instruction registers summary on page 255
Updated accessibility code	A.4.1 MIDR_EL1, Main ID Register on page 264
Updated accessibility code	A.4.2 MPIDR_EL1, Multiprocessor Affinity Register on page 265
Added fields description, and updated accessibility code	A.4.3 REVIDR_EL1, Revision ID Register on page 267
Changed description of all bits to 'Reserved'	A.4.12 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0 on page 285
Updated BF16 field description	A.4.15 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1 on page 291
Updated accessibility code	A.4.19 CCSIDR_EL1, Current Cache Size ID Register on page 302
Updated accessibility code	A.4.20 CLIDR_EL1, Cache Level ID Register on page 304
Updated accessibility code	A.4.22 CSSELR_EL1, Cache Size Selection Register on page 310
Updated accessibility code	A.4.23 CTR_EL0, Cache Type Register on page 312
Updated accessibility code	A.4.24 DCZID_EL0, Data Cache Zero ID register on page 314
Updated accessibility code	A.4.26 IMP_CPUPPMPDPCR_EL1, Global PPMPDP Configuration Register on page 318
Added RAZ field, and updated accessibility code	A.5.1 PMMIR_EL1, Performance Monitors Machine Identification Register on page 321

Change	Location
Updated reset values, fields descriptions, and accessibility code	A.5.2 PMCR_EL0, Performance Monitors Control Register on page 323
Added description for bits[31:0]	A.6.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Registers on page 349
Added ICV_AP0R0_EL1 register to the AArch64 GIC registers	A.6.4 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers on page 351
Added description for bits[31:0]	A.6.5 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Registers on page 353
Added ICV_AP1R0_EL1 register to the AArch64 GIC registers	A.6.6 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers on page 355
Updated accessibility code	A.7.1 TRBIDR_EL1, Trace Buffer ID Register on page 364
Added description to SEL field	A.8.2 ERRSELR_EL1, Error Record Select Register on page 369
Added description to DE field	A.8.3 ERXFR_EL1, Selected Error Record Feature Register on page 371
Added description to ER and SERR fields	A.8.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register on page 380
Updated PADDR field description	A.8.6 ERXADDR_EL1, Selected Error Record Address Register on page 388
Updated MV and AV fields description	A.8.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register on page 390
Added OF and SYN fields, removed CDNEN field	A.8.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register on page 396
Added description to all fields	A.8.10 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0 on page 404
Updated all registers accessibility code	A.10 AArch64 Trace registers summary on page 431
Updated EXLEVEL_S_EL2 field description indicating value `0b1` meaning	A.10.5 TRCIDR3, ID Register 3 on page 437
Changed description of all bits to 'Reserved'	A.10.14 TRCAUXCTLR, Auxiliary Control Register on page 449
Added AArch64 MPAM registers	A.11 AArch64 Memory Partitioning and Monitoring registers summary on page 461
Updated the description of some registers	B. External registers on page 468
Removed External Cluster registers	B. External registers on page 468
Removed External CTI registers	B. External registers on page 468
Updated EX field description indicating value `0b1` meaning, and updated N field value to six	B.2.27 PMCFGR, Performance Monitors Configuration Register on page 501
Changed description of all bits to 'Reserved'	B.3.2 EDACR, External Debug Auxiliary Control Register on page 543
Changed description of all bits to 'Reserved'	B.5.1 TRCAUXCTLR, External Auxiliary Control Register on page 598
Updated EXLEVEL_S_EL2 field description indicating value `0b1` meaning	B.5.12 TRCIDR3, External ID Register 3 on page 613

Change	Location
Removed the following registers from the external ROM table registers: • PRIDR0 • IICTRL • CLAIMSET • CLAIMCLR • DEVAFF0 • DEVAFF1 • LAR • LSR • DEVID2 • DEVID1 • DEVID • DEVTYPE • PIDR5 • PIDR6 • PIDR7	B.6 External ROM table registers summary on page 650
Removed POWERID field value, and indicated POWERIDVALID field value as `0b0`	B.6.1 ROMENTRY0, Class 0x9 ROM Table Entries on page 651
Removed POWERID field value, and indicated POWERIDVALID field value as `0b0`	B.6.2 ROMENTRY1, Class 0x9 ROM Table Entries on page 653
Removed POWERID field value, and indicated POWERIDVALID field value as `0b0`	B.6.3 ROMENTRY2, Class 0x9 ROM Table Entries on page 655
Added POWERID field and description	B.6.4 ROMENTRY3, Class 0x9 ROM Table Entries on page 657

Table C-4: Differences between issue 0000-04 and issue 0100-05

Change	Location
First Confidential early access release for r1p0	-
Editorial changes	Throughout document
Theodul name changed to DSU-110	Throughout document
Replaced ETM with ETE or 'trace unit' where required	Throughout document
• Added *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile*. This document includes Embedded Trace Extension description. • Updated SVE document name	Additional reading
Updated description and removed the number of cores in a cluster	2.1 Cortex-A715 core features on page 22
Updated title	2.2 Cortex-A715 core configuration options on page 24
• Renamed features as per 'Feature names for A-profile' page in developer.arm.com website • Added FEAT_SPEv1p1, FEAT_Debugv8p4, FEAT_DoPD	2.4 Supported standards and specifications on page 25
Updated the descriptions	2.6 Design tasks on page 29

Change	Location
Updated L1 data memory page size description (4KB, 16KB, 64KB, 2MB)	3.1 Core components on page 31
Updated the core voltage domains and power domains figure	5.1 Voltage and power domains on page 38
Updated content	5.2.2 Low-power state behavior considerations on page 41
Added details and a Note	5.4.2 Off mode on page 44
Updated description of Full retention mode exit conditions	5.4.4 Full retention mode on page 44
Added information in the powerdown sequence	5.6 Cortex-A715 core powerup and powerdown sequence on page 48
L2 TLB Medium page, corrected typo in table (521MB instead of 512MB)	6.1 Memory Management Unit components on page 51
Updated last sentence	6.4 Translation table walks on page 54
• Updated 'External aborts' description • Updated 'Conflict aborts' description	6.6 Responses on page 55
• Throughout the chapter, added links to the registers instead of providing tables with redundant for the returned data. Updated descriptions information • Added more details in fields description	10. Direct access to internal memory on page 71
• Updated the note • Updated description for events 0x0020, 0x0029, 0x002A, 0x002B, 0x0031, 0x0036, 0x0037, 0x0050, 0x0051, 0x0052, 0x0053, 0x0056, 0x0057, 0x0058, 0x0059, 0x0060, 0x0061, 0x400D, 0x400E, 0x400F, 0x4010, 0x4011, 0x4012, 0x4013 • Added event 0x8079	18.1 Performance monitors events on page 98
Updated the explanation considering the trace unit throughout the chapter	19. Embedded Trace Extension support on page 113
Updated External memory-mapped access topic	21.1 Activity monitors access on page 123
Updated events AMEVCNTR10, AMEVCNTR11, and AMEVCNTR12 name and description	21.3 Activity monitors events on page 124
Added reset values to all registers	A. AArch64 registers on page 130
Added description for bits[55], [54:53], [52:47], and [40]	A.1.15 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 159
Added bits description	A.1.22 IMP_ISIDE_DATA0_EL3, RAMINDEX Instruction Data register 0 on page 185
Added bits description	A.1.23 IMP_ISIDE_DATA1_EL3, RAMINDEX Instruction Data register 1 on page 187
Added bits description	A.1.24 IMP_ISIDE_DATA2_EL3, RAMINDEX Instruction Data register 2 on page 189
Added bits description	A.1.25 IMP_MMU_DATA0_EL3, RAMINDEX TLB Data register 0 on page 191
Added bits description	A.1.26 IMP_MMU_DATA1_EL3, RAMINDEX TLB Data register 1 on page 195
Added bits description	A.1.27 IMP_MMU_DATA2_EL3, RAMINDEX TLB Data register 2 on page 201
Added bits description	A.1.28 IMP_DSIDE_DATA0_EL3, RAMINDEX L1D Data register 0 on page 206

Change	Location
Added bits description	A.1.29 IMP_DSIDE_DATA1_EL3, RAMINDEX L1D Data register 1 on page 208
Added bits description	A.1.30 IMP_DSIDE_DATA2_EL3, RAMINDEX L1D Data register 2 on page 210
Added bits description	A.1.31 IMP_L2_DATA0_EL3, RAMINDEX L2 Data register 0 on page 212
Updated bits description	A.1.32 IMP_L2_DATA2_EL3, RAMINDEX L2 Data register 2 on page 216
Updated bits description	A.1.33 IMP_L2_DATA1_EL3, RAMINDEX L2 Data register 1 on page 218
Updated bits description	A.1.43 FPCR, Floating-point Control Register on page 235
Added bits description	A.3.1 RAMINDEX, RAMINDEX system instruction on page 255
Updated bits description for r1p0	A.4.1 MIDR_EL1, Main ID Register on page 264
Updated bits description	A.4.7 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0 on page 273
Added description for bits[35:12]	A.4.8 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1 on page 276
Updated bits description	A.4.9 ID_AA64ZFR0_EL1, SVE Feature ID register 0 on page 278
Updated description for bits[11:8]	A.4.16 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0 on page 294
Added description for bits[61:32], and [27:24]	A.4.17 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1 on page 296
Changed FEAT_MTE to FEAT_MTE2	A.4.19 CCSIDR_EL1, Current Cache Size ID Register on page 302
Added description for bits[19:8]	A.5.1 PMMIR_EL1, Performance Monitors Machine Identification Register on page 321
Updated bits description	A.5.2 PMCR_EL0, Performance Monitors Control Register on page 323
Updated description	A.5.3 PMCEID0_EL0, Performance Monitors Common Event Identification register 0 on page 327
Updated description	A.5.4 PMCEID1_EL0, Performance Monitors Common Event Identification register 1 on page 334
Updated bits description	A.6.2 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register on page 346
Added description for bit[28]	A.8.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register on page 390
Updated description for bits[18:6], and [31:28]	A.8.10 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0 on page 404
Updated description for bits[7:0]	A.9.4 AMCFGR_EL0, Activity Monitors Configuration Register on page 422
Updated description for bit[31]	A.10.4 TRCIDR2, ID Register 2 on page 435
Bit[23], LPOVERRIDE 0b0 value described	A.10.7 TRCIDR5, ID Register 5 on page 441
Updated description for bits[3:0]	A.10.16 TRCIDR1, ID Register 1 on page 452
Updated bits[19:16] description	A.10.17 TRCDEVARCH, Device Architecture Register on page 453
Added reset values to all registers	B. External registers on page 468
Updated bits description	B.2.27 PMCFGR, Performance Monitors Configuration Register on page 501

Change	Location
Updated description for bit[2]	B.2.28 PMCR_EL0, External Performance Monitors Control Register on page 503
Updated description	B.2.29 PMCEID0, Performance Monitors Common Event Identification register 0 on page 507
Updated description	B.2.30 PMCEID1, Performance Monitors Common Event Identification register 1 on page 511
Updated description	B.2.31 PMCEID2, Performance Monitors Common Event Identification register 2 on page 515
Updated description	B.2.32 PMCEID3, Performance Monitors Common Event Identification register 3 on page 519
Updated description for bits[7:4]	B.2.40 PMPIDR2, Performance Monitors Peripheral Identification Register 2 on page 532
Updated bits description for r1p0	B.3.4 MIDR_EL1, External Main ID Register on page 547
Updated description for bits[7:4]	B.3.15 EDPIDR2, External Debug Peripheral Identification Register 2 on page 564
Updated description for bits[7:0]	B.4.9 AMCFGR, Activity Monitors Configuration Register on page 582
Updated bits description for r1p0	B.4.10 AMIIDR, Activity Monitors Implementation Identification Register on page 584
Updated description for bits[7:4]	B.4.16 AMPIDR2, Activity Monitors Peripheral Identification Register 2 on page 591
Updated bits description for TRCSTALLCTLR, Stall Control Register	B.4.2 TRCSTALLCTLR, Stall Control Register on page 520
Updated description for bits[3:0]	B.5.10 TRCIDR1, External ID Register 1 on page 610
Updated description for bit[31]	B.5.11 TRCIDR2, External ID Register 2 on page 611
Bit[23], LPOVERRIDE 0b0 value described	B.5.14 TRCIDR5, External ID Register 5 on page 617
Updated bits[19:16] description	B.5.20 TRCDEVARCH, External Device Architecture Register on page 627
Updated description for bits[7:4]	B.5.31 TRCPIDR2, Peripheral Identification Register 2 on page 642
Updated bits description	B.5.32 TRCPIDR3, Peripheral Identification Register 3 on page 644

Table C-5: Differences between issue 0100-05 and issue 0101-06

Change	Location
First Confidential early access release for r1p1	-
Editorial changes	Throughout document
Added FEAT_ECBHB	2.4 Supported standards and specifications on page 25
Added revision r1p1	2.7 Product revisions on page 30
Updated description	5.5.1 Maximum Power Mitigation Mechanism on page 46
Added a new topic	5.5.3 Dispatch block on page 48

Change	Location
Updated sequence description	5.6 Cortex-A715 core powerup and powerdown sequence on page 48
Added a new topic	5.6.1 Managing RAS fault and error interrupts during the core powerdown on page 49
Added a new topic	6.8 Page-based hardware attributes on page 58
Updated the section on 'Hardware data prefetcher'	8.5 Data prefetching on page 66
Updated description	9.2 Support for memory types on page 69
Updated description	10. Direct access to internal memory on page 71
Added <n> value for L2 = 128KB	10.2 L2 cache encodings on page 74
Updated SECDED ECC description	11.1 Cache protection behavior on page 77
Updated decription	11.2 Error containment on page 78
Updated 'Uncontainable errors' section description	11.5 Error injection on page 80
Added chapter	12. Utility bus on page 82
Updated Peripheral ID value and Core revision	17.6 CoreSight component identification on page 94
Added following registers: IMP_CPUPSELR_EL3, IMP_CPUPCR_EL3, IMP_CPUPOR_EL3, IMP_CPUPMR_EL3, IMP_CPUPOR2_EL3, IMP_CPUPMR2_EL3, IMP_CPUPFR_EL3	A.1 AArch64 Generic System Control registers summary on page 130
Updated reset values for register LORID_EL1	A.1 AArch64 Generic System Control registers summary on page 130
Added 'Access' topic to registers: AMAIR_EL2, AMAIR_EL1, IMP_ATCR_EL2, IMP_ATCR_EL1, AFSR0_EL2, AFSR1_EL2, AFSR0_EL1, AFSR1_EL1	A.1 AArch64 Generic System Control registers summary on page 130
Updated SLCID_IDX field description	A.1.34 IMP_CLUSTERCDBG_EL3, Cluster Cache Debug Register on page 219
Updated some fields description	A.2.1 IMP_CPUPPMCR_EL3, Global PPM Configuration Register on page 253
Updated reset values for registers: MIDR_EL1, ID_AA64PFR0_EL1, ID_AA64PFR1_EL1, ID_AA64ZFR0_EL1, ID_AA64DFR0_EL1, ID_AA64ISAR0_EL1, ID_AA64ISAR1_EL1, ID_AA64MMFR0_EL1, ID_AA64MMFR1_EL1, ID_AA64MMFR2_EL1, GMID_EL1, DCZID_EL0, MPAMIDR_EL1	A.4 AArch64 Identification registers summary on page 263

Change	Location
Added 'Access' topic to register CCSIDR_EL1	A.4 AArch64 Identification registers summary on page 263
Updated some fields description	A.4.1 MIDR_EL1, Main ID Register on page 264
Updated CSV2_frac description	A.4.8 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1 on page 276
Updated some fields description	A.4.15 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1 on page 291
Updated some fields description	A.4.16 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0 on page 294
Updated reset values and some fields description	A.5.1 PMMIR_EL1, Performance Monitors Machine Identification Register on page 321
Updated reset values, added IDCODE field, removed X field	A.5.2 PMCR_EL0, Performance Monitors Control Register on page 323
Added 'Access' topic to registers: ICC_AP0R0_EL1, ICV_AP0R0_EL1, ICC_AP1R0_EL1, ICV_AP1R0_EL1	A.6 AArch64 GIC system registers summary on page 341
Updated reset values	A.7.1 TRBIDR_EL1, Trace Buffer ID Register on page 364
Updated reset values for registers: ERRIDR_EL1, ERXFR_EL1, ERXPFGF_EL1	A.8 AArch64 RAS registers summary on page 367
Added 'Access' topic to registers: ERXFR_EL1, ERXCTLR_EL1, ERXSTATUS_EL1, ERXADDR_EL1, ERXPFGF_EL1, ERXPFGCTL_EL1, ERXPFGCDN_EL1, ERXMISC0_EL1, ERXMISC1_EL1, ERXMISC2_EL1, ERXMISC3_EL1	A.8 AArch64 RAS registers summary on page 367
Updated some fields description	A.8.3 ERXFR_EL1, Selected Error Record Feature Register on page 371
Updated some fields description	A.8.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register on page 380
Updated most fields description	A.8.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register on page 390

Change	Location
Added field CDNEN, removed fields SYN , MV, AV, PN, ER, CI, OF	A.8.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register on page 396
Updated reset values for registers: AMEVTYPER10_EL0, AMEVCNTR11_EL0, AMEVCNTR12_EL0, AMCFGR_EL0, AMEVCNTR00_EL0, AMEVCNTR01_EL0, AMEVCNTR02_EL0, AMEVCNTR03_EL0	A.9 AArch64 Activity Monitors registers summary on page 417
Added 'Access' topic to registers: AMEVTYPER10_EL0, AMEVCNTR11_EL0, AMEVCNTR12_EL0, AMEVCNTR00_EL0, AMEVCNTR01_EL0, AMEVCNTR02_EL0, AMEVCNTR03_EL0	A.9 AArch64 Activity Monitors registers summary on page 417
Updated reset values for registers: TRCIDR8, TRCIMSPEC0, TRCIDR2, TRCIDR3, TRCIDR4, TRCIDR5, TRCIDR0, TRCIDR1	A.10 AArch64 Trace registers summary on page 431
Added 'Access' topic to register TRCAUXCTLR	A.10 AArch64 Trace registers summary on page 431
Updated description for fields SYSSTALL, STALLCTL	A.10.5 TRCIDR3, ID Register 3 on page 437
Updated some fields description	A.10.16 TRCIDR1, ID Register 1 on page 452
Added External MPMM registers	B.1 External MPMM registers summary on page 468
Added a set of Snapshot registers: from PMPCSSR (Offset `0x600`) to PMSSCR (Offset `0x6F0`)	B.2 External PMU registers summary on page 474
Updated reset values for registers: PMCFGR, PMCR_EL0, PMMIR, PMDEVID, PMPIDR4, PMPIDR0, PMPIDR1, PMPIDR2, PMPIDR3, PMCIDR1	B.2 External PMU registers summary on page 474
Added 'Accessibility' topic to registers: PMCFGR, PMCR_EL0, PMCEID0, PMCEID1, PMCEID2, PMCEID3, PMMIR, PMDEVARCH, PMDEVID, PMDEVTYPE, PMPIDR0, PMPIDR1, PMPIDR2, PMPIDR3, PMPIDR4, PMCIDR0, PMCIDR1, PMCIDR2, PMCIDR3,	B.2 External PMU registers summary on page 474
Updated description for fields EX, N	B.2.27 PMCFGR, Performance Monitors Configuration Register on page 501
Replaced X field with **RAZ/WI**	B.2.28 PMCR_EL0, External Performance Monitors Control Register on page 503
Updated reset values for registers: MIDR_EL1, EDPFR, EDDFR, EDDEVARCH, EDDEVID1, EDDEVID, EDPIDR4, EDPIDR0, EDPIDR1, EDPIDR2, EDPIDR3, EDCIDR1	B.3 External Debug registers summary on page 541
Added 'Accessibility' topic to registers: EDRCR, EDACR, EDPRCR, MIDR_EL1, EDPFR, EDDFR, EDDEVARCH, EDDEVID2, EDDEVID1, EDDEVID, EDDEVTYPE, EDPIDR4, EDPIDR0, EDPIDR1, EDPIDR2, EDPIDR3, EDCIDR0, EDCIDR1, EDCIDR2, EDCIDR3	B.3 External Debug registers summary on page 541
Updated some fields description	B.3.3 EDPRCR, External Debug Power/Reset Control Register on page 545

Change	Location
Updated some fields description	B.3.4 MIDR_EL1, External Main ID Register on page 547
• Register has 2 offsets to access 64 bits: 0xD20 provides access to lower 32 bits, 0xD24 to upper 32 bits) • Updated some fields description	B.3.5 EDPFR, External Debug Processor Feature Register on page 548
Register has 2 offsets to access 64 bits: 0xD28 provides access to lower 32 bits, 0xD2C to upper 32 bits)	B.3.6 EDDFR, External Debug Feature Register on page 551
Updated some fields description	B.3.16 EDPIDR3, External Debug Peripheral Identification Register 3 on page 565
Updated reset values for registers: AMCGCR, AMCFGR, AMIIDR, AMDEVARCH, AMPIDR3, AMCIDR1	B.4 External AMU registers summary on page 572
Updated some fields description	B.4.10 AMIIDR, Activity Monitors Implementation Identification Register on page 584
Updated some fields description	B.4.17 AMPIDR3, Activity Monitors Peripheral Identification Register 3 on page 592
Removed TRCSTALLCTLR register that is not implemented	B.5 External ETE registers summary on page 597
Updated reset values for registers: TRCIDR8, TRCIMSPEC0, TRCIDR0, TRCIDR1, TRCIDR2, TRCIDR3, TRCIDR4, TRCIDR5, TRCDEVTYPE, TRCPIDR3, TRCCIDR1	B.5 External ETE registers summary on page 597
Added 'Accessibility' topic to registers: TRCAUXCTLR, TRCIDR8, TRCIDR9, TRCIDR10, TRCIDR11, TRCIDR12, TRCIDR13, TRCIMSPEC0, TRCIDR0, TRCIDR1, TRCIDR2, TRCIDR3, TRCIDR4, TRCIDR5, TRCIDR6, TRCIDR7, TRCITCTRL, TRCCLAIMSET, TRCCLAIMCLR, TRCDEVARCH, TRCDEVID2, TRCDEVID1, TRCDEVID, TRCDEVTYPE, TRCPIDR4, TRCPIDR5, TRCPIDR6, TRCPIDR7, TRCPIDR0, TRCPIDR1, TRCPIDR2, TRCPIDR3, TRCCIDR0, TRCCIDR1, TRCCIDR2, TRCCIDR3	B.5 External ETE registers summary on page 597
Updated some fields description	B.5.12 TRCIDR3, External ID Register 3 on page 613
Updated some fields description	B.5.32 TRCPIDR3, Peripheral Identification Register 3 on page 644
Updated reset values for registers: ROMENTRY3, DEVARCH, PIDR4, PIDR0, PIDR1, PIDR2, PIDR3, CIDR0, CIDR1, CIDR2, CIDR3	B.6 External ROM table registers summary on page 650
Updated some fields description	B.6.4 ROMENTRY3, Class 0x9 ROM Table Entries on page 657
Updated some fields description	B.6.10 PIDR2, Peripheral Identification Register 2 on page 664
Updated some fields description	B.6.11 PIDR3, Peripheral Identification Register 3 on page 665

Table C-6: Differences between issue 0101-06 and issue 0101-07

Change	Location
Second early access release for r1p1	-
Updated product name	Throughout document

Table C-7: Differences between issue 0101-07 and issue 0102-08

Change	Location
First release for r1p2	-
Editorial changes	Throughout document
Added revision r1p2	2.7 Product revisions on page 30
Changed topic title 'Additional reading' to 'Useful resources'	1.4 Useful resources on page 20
Added chapter	17.7 CTI register identification values on page 94
Updated Hardware data prefetcher description	8.5 Data prefetching on page 66
Updated Aarch64 registers 'Accessibility' topic: AFSR0_EL1, AFSR0_EL2, AFSR1_EL1, AFSR1_EL2, AMAIR_EL2, ERXADDR_EL1, ERXFR_EL1, ERXMISC0_EL1, ERXMISC2_EL1, ERXMISC3_EL1, ERXPFGCDN_EL1, ERXPFGCTL_EL1, ERXPFGF_EL1, ERXSTATUS_EL1, IMP_ATCR_EL1, IMP_ATCR_EL2,	A.1.46 AFSR0_EL1, Auxiliary Fault Status Register 0 (EL1) on page 244, A.1.44 AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2) on page 239, A.1.47 AFSR1_EL1, Auxiliary Fault Status Register 1 (EL1) on page 247, A.1.45 AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2) on page 242, A.1.6 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2) on page 141, A.8.6 ERXADDR_EL1, Selected Error Record Address Register on page 388, A.8.3 ERXFR_EL1, Selected Error Record Feature Register on page 371, A.8.10 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0 on page 404, A.8.12 ERXMISC2_EL1, Selected Error Record Miscellaneous Register 2 on page 412, A.8.13 ERXMISC3_EL1, Selected Error Record Miscellaneous Register 3 on page 415, A.8.9 ERXPFGCDN_EL1, Selected Pseudo-fault Generation Countdown register on page 400, A.8.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register on page 396, A.8.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register on page 390, A.8.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register on page 380, A.1.21 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register on page 182, A.1.19 IMP_ATCR_EL2, CPU Auxiliary Translation Control Register on page 175
Updated External registers 'Accessibility' topic: CPUPPMPDPCR, EDDFR, EDPFR	B.1.3 CPUPPMPDPCR, Global PPMPDP Configuration Register on page 472, B.3.6 EDDFR, External Debug Feature Register on page 551, B.3.5 EDPFR, External Debug Processor Feature Register on page 548
Updated External register CPUPPMPDPCR description	B.1 External MPMM registers summary on page 468
Updated External registers EDPFR and EDDFR description	B.3 External Debug registers summary on page 541