Arm® Neoverse™ N2 Core

Revision: r0p0

Technical Reference Manual

Non-Confidential

Issue 04

102099_0000_04_en

Arm® Neoverse™ N2 Core
Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-02	16 October 2020	Non-Confidential	First early access release for r0p0
0000-03	22 February 2021	Confidential	Second early access release for r0p0
0000-04	25 May 2021	Non-Confidential	Third early access release for r0p0

Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

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.

Web Address

developer.arm.com

Progressive terminology commitment

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

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

If you find offensive terms in this document, please contact terms@arm.com.

Contents

1 Introduction

1.1 Product revision status

The `rxpy` identifier indicates the revision status of the product described in this manual, for example, `r1p2`, where:

`rx` Identifies the major revision of the product, for example, r1.

`py` 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.

Typographic conventions

Convention	Use
italic	Introduces citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace bold`	Denotes language keywords when used outside example code.
`monospace underline`	Denotes 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>`

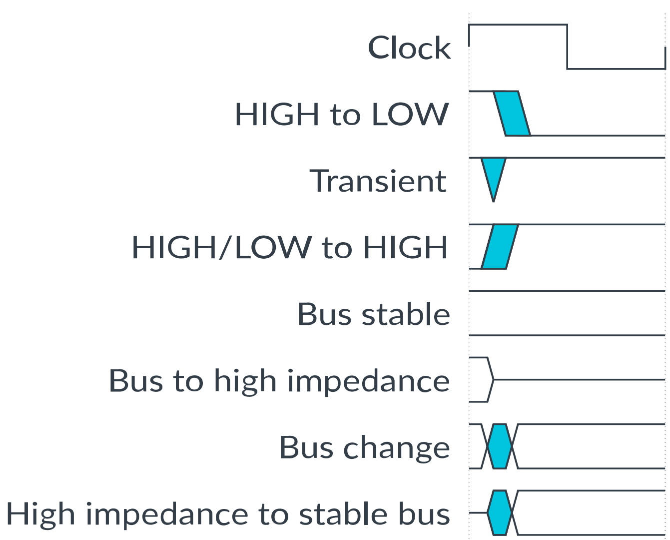
Convention	Use
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.
Caution	This represents a recommendation which, if not followed, might lead to system failure or damage.
Warning	This represents a requirement for the system that, if not followed, might result in system failure or damage.
Danger	This represents a requirement for the system that, if not followed, will result in system failure or damage.
Note	This represents an important piece of information that needs your attention.
Tip	This represents a useful tip that might make it easier, better or faster to perform a task.
Remember	This is a reminder of something important that relates to the information you are reading.

Timing diagrams

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

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

Figure 1-1: Key to timing diagram conventions



Signals

The signal conventions are:

Signal level

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

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

Lowercase n

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

1.4 Additional reading

This document contains information that is specific to this product. See the following documents for other relevant information:

Table 1-2: Arm publications

Document Name	Document ID	Licensee only
Neoverse™ N2 Release Note	-	Yes
Arm® Neoverse™ N2 Core Cryptographic Extension Technical Reference Manual	102101	No
Arm® Neoverse™ N2 Core Configuration and Integration Manual	102100	Yes
Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual	101381	Yes
Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual	101382	Yes
Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile	DDI 0487	No
Arm® Embedded Trace Extension	ARM-ECM-0721850	Yes
Arm® Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM) for Armv8-A	DDI 0598	No
Arm®v9-A Supplement for v8-A Arm® Architecture Reference Manual	DDI 0608	No
Arm® Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile	DDI 0587	No
Arm® Architecture Reference Manual Supplement The Scalable Vector Extension (SVE) for Armv8-A	DDI 0584	No
AMBA® 5 CHI Architecture Specification	IHI 0050	No

Document Name	Document ID	Licensee only
Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4	IHI 0069	No
Arm® CoreSight™ Architecture Specification v3.0	IHI 0029	No
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual	101089	No

Table 1-3: Other publications

Document ID	Document Name
-	-

1.5 Feedback

Arm welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

Information about how to give feedback on the content.

If you have comments on content then send an e-mail to errata@arm.com. Give:

- The title Arm® Neoverse™ N2 Core Technical Reference Manual.
- The number 102099_0000_04_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

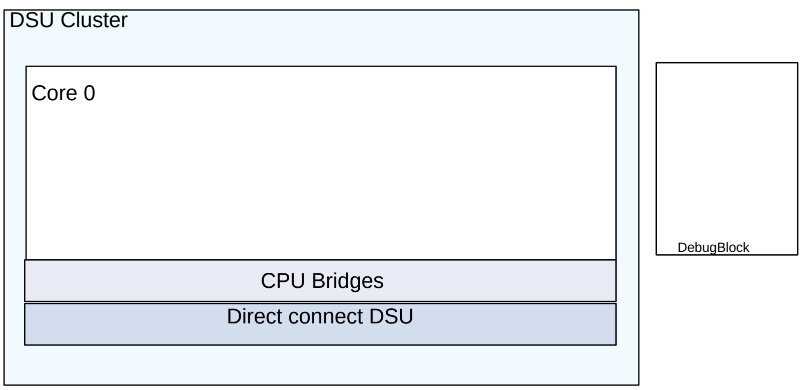
2 The Neoverse™ N2 core

The Neoverse™ N2 core is a high-performance and low-power product that implements the Arm®v9.0-A architecture. This implementation supports all previous Armv8-A architecture implementations up to and including Arm®v8.5-A.

The Neoverse™ N2 core is implemented inside a DSU-110 cluster.

The following figure shows an example configuration with one Neoverse™ N2 core that is implemented as a single core in a DSU cluster which is configured for Direct connect, without the L3 cache, snoop filter, or *Snoop Control Unit* (SCU) logic present. The Neoverse™ N2 core supports Direct connect only. For more information on the DSU Direct connect, see the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*.

Figure 2-1: Neoverse™ N2 example configuration



This manual applies to the Neoverse™ N2 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 Armv8, for Armv8-A architecture profile*.

2.1 Neoverse™ N2 core features

The Neoverse™ N2 core should be used in standalone DSU configurations, that is configured in Direct connect mode.

Core features

- Implementation of the Armv9-A A32, T32, and A64 instruction sets
- AArch32 Execution state at Exception level EL0 and AArch64 Execution state at all Exception levels, EL0 to EL3
- *Memory Management Unit* (MMU)
- 48-bit *Physical Address* (PA) and 48-bit *Virtual Address* (VA)

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

Cache features

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

Debug features

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

2.2 Neoverse™ N2 core implementation options

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

The Neoverse™ N2 core implementation options include:

L2 cache size
: You can configure the L2 cache to be 512KB or 1024KB.

L2 transaction queue size
: You can configure the L2 transaction queue size to be 48, 56, or 64.

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

Coherent Instruction Cache

You can configure your implementation with or without support for coherent Instruction Cache.

Random Number Generator

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

CoreSight *Embedded Logic Analyzer* (ELA)

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

Size of the ATB FIFO depth in the core ELA

You can configure the size of the ATB FIFO to be 4, 8, 16, 32, or 64.

Timing closure

You can configure the L2 data cache RAMs timing behavior.

2.3 DSU-110 dependent features

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

The following table describes which DSU-110 dependent features are supported in your Neoverse™ N2 core.

Table 2-1: Neoverse™ N2 core features that have a dependency on the DSU-110

Feature	Supported in the Neoverse™ N2 core	Dependency on the DSU-110
Direct connect	Only supports Direct connect	Direct connect support at the cluster level only applies when your licensed core also supports Direct connect. Direct connect is intended for large systems where there are many cores.
Core included in a complex	No	Affects the cluster configuration and external signals.
Cryptographic Extension	Yes	Affects the external signals of the DSU-110.
Maximum Power Mitigation Mechanism (MPMM)	Yes	This affects the external signals of the DSU-110.
Performance Defined Power (PDP) feature	Yes	Affects the external signals of the DSU-110.
DISPBLKx signal supported	Yes	Affects the external signals of the DSU-110.
Statistical Profiling Extension (SPE) architecture	Yes	Affects the external signals of the DSU-110.
Physical Address (PA) width	48-bit	Affects the CHI master port bus width. For more details, see the following parts of the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*: • *CHI master interface*

Note

The Cryptographic Extension is supplied under a separate license.

2.4 Supported standards and specifications

The Neoverse™ N2 core implements the Arm®v9.0-A architecture and supports all previous Armv8-A architectures up to Arm®v8.5-A. It also implements specific Arm architecture extensions and supports interconnect, interrupt, timer, debug, and trace architectures.

The Neoverse™ N2 core supports AArch32 at EL0 and AArch64 at all Exception levels, EL0 to EL3, and supports all mandatory features of each architecture version up to Arm®v8.5-A.

Note

Arm®v8.6-A is completely optional for Arm®v9.0-A implementations.

The following tables show, for each Armv8-A architecture version, the optional features that the Neoverse™ N2 core supports.

Table 2-2: Armv8.0-A optional feature support in the Neoverse™ N2 core

Feature	Status	Notes
Cryptographic Extension	Supported using a configurable option	See the *Arm® Neoverse™ N2 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 13 Advanced SIMD and floating-point support on page 92 for more technical reference and register information.
Performance Monitoring Extension (PME)	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on this feature.

Table 2-3: Arm®v8.1-A optional feature support in the Neoverse™ N2 core

Feature	Status	Notes
Armv8.1-TTHM, Hardware management of the Access flag and dirty state	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.1-VMID16, 16-bit VMID	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.

Table 2-4: Arm®v8.2-A optional feature support in the Neoverse™ N2 core

Feature	Status	Notes
Armv8.2-TTPBHA, Translation Table Page-Based Hardware Attributes	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-PCSample, PC Sample-based profiling	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-SHA, SHA2-512 and SHA3 functionality	Supported as part of Armv8-A Cryptographic Extension	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-VPIPT, VMID-aware PIPT instruction cache	Not supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-SM, SM3 and SM4 functionality	Supported as part of Armv8-A Cryptographic Extension	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-BF16, 16-bit floating-point instructions	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-I8MM, Int8 Matrix Multiply instructions	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-AA32BF16, 16-bit floating-point instructions	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-AA32I8MM, Int8 Matrix Multiply instructions	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-F32MM, FP32 Matrix Multiply instructions	Not supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-F64MM, FP64 Matrix Multiply instructions	Not supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Scalable Vector Extension (SVE)	Supported	See 14 Scalable Vector Extensions support on page 93 and the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on this extension.
Armv8.2-LPA, Large *Physical Address* (PA) and *Intermediate PA* (IPA) support	Not supported	-
Armv8.2-LVA, Large *Virtual Address*(VA) support	Not supported	-
Armv8.2-LSMAOC, Load/ Store Multiple Atomicity and Ordering Controls	Not supported	-
Armv8.2-AA32HPD, AArch32 Hierarchical Permission Disables	Not supported	-
Statistical Profiling Extension (SPE)	Supported	-

Table 2-5: Arm®v8.3-A optional feature support in the Neoverse™ N2 core

Feature	Status	Notes
Armv8.3-NV, Nested Virtualization	Supported	-
Armv8.3-CCIDX, Cache extended number of sets	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on this feature.

Feature	Status	Notes
Armv8.3-PAuth2, Pointer Authentication enhancements	Supported	-
Armv8.3-FPAC, *Faulting Pointer Authentication Code* (FPAC)	Supported	-

Table 2-6: Arm®v8.4-A optional feature support in the Neoverse™ N2 core

Feature	Status	Notes
Activity Monitors Extension	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on this feature.
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.
Armv8.4-NV, enhanced support for Nested Virtualization	Supported	-

Table 2-7: Arm®v8.5-A optional feature support in the Neoverse™ N2 core

Feature	Status	Notes
Arm8.5-MemTag, *Memory Tagging Extension* (MTE)	Supported	The Neoverse™ N2 core always implements MTE and therefore is compliant with the CHI Issue 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.
Armv8.5-RNG, Random Number Generator instructions	Supported using a configurable option	-
Armv8.5-CSEH, Context Synchronization and Exception Handling	Not supported	-

Table 2-8: Arm®v8.6-A optional feature support in the Neoverse™ N2 core

Feature	Status	Notes
Armv8.6-DGH, *Data Gathering Hint*	Supported	Adds the Data Gathering Hint instruction to the hint space.
Armv8.6-ECV, *Enhanced Counter Virtualization*	Not supported	-
Armv8.6-FGT, *Fine Grain Traps*	Not supported	-
Armv8.6-TWED, *Delayed Trapping of WFE*	Not supported	-
Armv8.6-AMU, *Activity Monitors Unit (AMU) Extension*	Not supported	-
Armv8.6-MPAM, *Memory Partitioning and Monitoring (MPAM) Extension*	Not supported	-
Armv8.6-MTPMU, *Multi-threaded PMU Extensions*	Not supported	-

The following table shows the Arm®v9.0-A features that the Neoverse™ N2 core supports.

Table 2-9: Arm®v9.0-A feature support in the Neoverse™ N2 core

Feature	Status	Notes
Scalable Vector Extension 2 (SVE2)	Supported	See 14 Scalable Vector Extensions support on page 93.
Embedded Trace Extension (ETE)	Supported	See 19 Embedded Trace Extension support on page 118.
Trace Buffer Extension (TRBE)	Supported	See 20 Trace Buffer Extension support on page 127.
SVE2 SM4 instructions	Supported	-
SVE2 SHA-3 instructions	Supported	
SVE2 bit permute instructions	Supported	
SVE2 AES instructions	Supported	
Transactional Memory Extension (TME)	Not supported	-

The following table shows the other standards and specifications that the Neoverse™ N2 core supports.

Table 2-10: Other standards and specifications support in the Neoverse™ N2 core

Standard or specification	Version	Notes
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 See the *Arm®v8.5 Debug Architecture* for information on this architecture.
CoreSight	v3.0	See the *Arm® CoreSight™ Architecture Specification v3.0* for more information.
Reliability, Availability, and Serviceability (RAS)	-	All extensions up to Arm®v9.0-A with *Error Correcting Code* (ECC) configured. See 11 RAS extension support on page 85 for more information on the implementation of this extension in the core.

2.5 Test features

The Neoverse™ N2 core provides test signals that enable the use of both *Automatic Test Pattern Generation* (ATPG) and *Memory Built-In Self Test* (MBIST) to test the core logic and memory arrays.

The Neoverse™ N2 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® Neoverse™ N2 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® Neoverse™ N2 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 Neoverse™ N2 core is delivered as a synthesizable RTL description in SystemVerilog. Before you can use the Neoverse™ N2 core, you must implement, integrate, and program it.

Separate parties can perform each of the following tasks. Implementation and integration choices affect the behavior and features of the core:

Implementation
: The implementer configures and synthesizes the RTL to produce a hard macrocell. This task includes integrating RAMs into the design.

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

Programming
: In the final task, the system programmer develops the software to configure and initialize the core and tests the application software.

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

Build configuration
: The implementer chooses the options that affect how the RTL source files are rendered. These options usually include or exclude logic that can affect the area, maximum frequency, and features of the resulting macrocell.

Configuration inputs
: 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. They can also limit the options available to the software.

Software configuration
: The programmer configures the core by programming values into registers. The configuration choices affect the behavior of the core.

See *RTL configuration process* in the *Arm® Neoverse™ N2 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.

Note

The *Arm® Neoverse™ N2 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.

2.7 Product revisions

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

Table 2-11: Product revisions

Revision	Notes
r0p0	First draft release

Changes in functionality that have an impact on the documentation also appear in E Document revisions on page 607.

3 Technical overview

All components in the Neoverse™ N2 core are always present. These components are designed to make the Neoverse™ N2 core a high-performance or balanced-performance core.

The main blocks include:

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

The Neoverse™ N2 core interfaces with the DSU-110 through the CPU bridge.

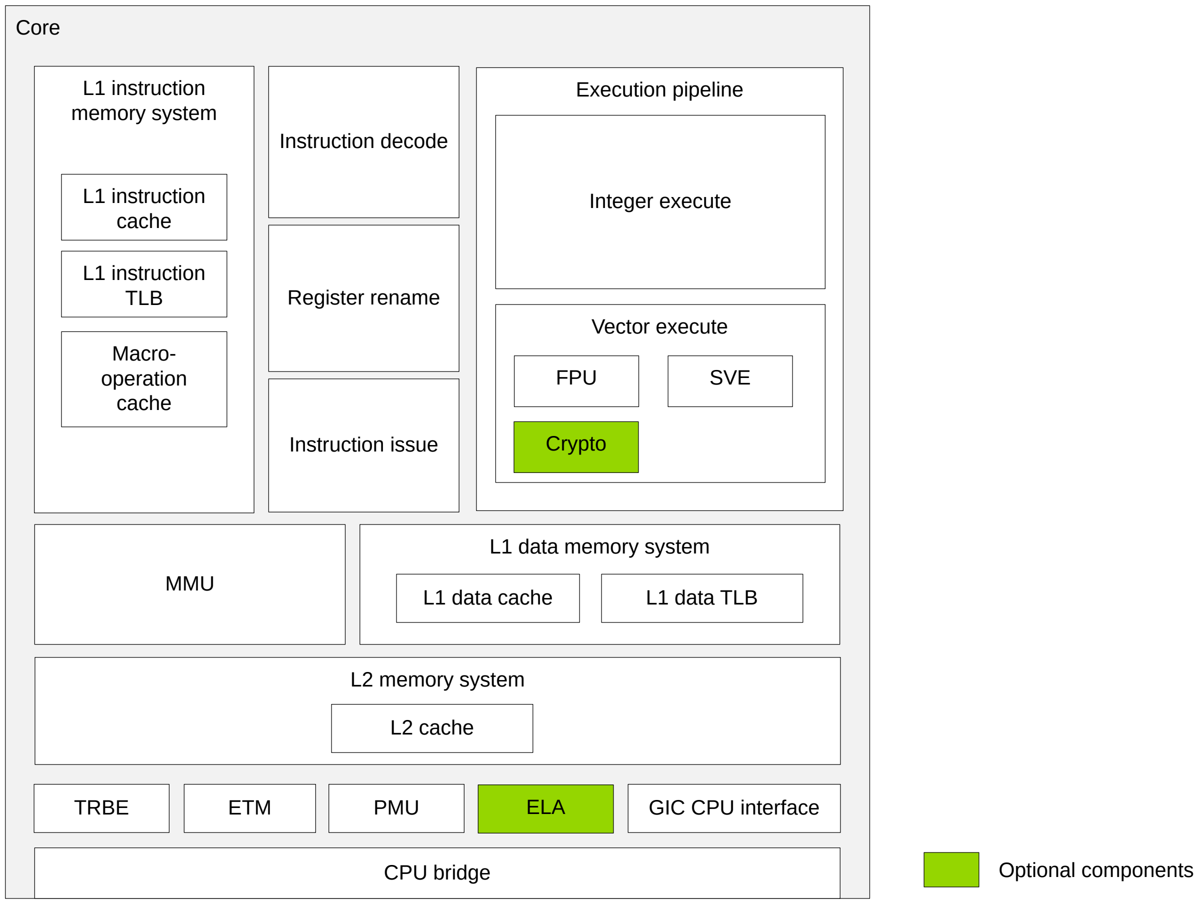
The Neoverse™ N2 core implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Armv8-A architectures up to Arm®v8.5-A. The programmers model and the architecture features implemented, such as the Generic Timer, are compliant with the standards in 2.4 Supported standards and specifications on page 26.

3.1 Core components

The Neoverse™ N2 core includes components designed to make it a high-performance and low-power product. The Neoverse™ N2 core includes a CPU bridge that connects the core to the DSU-110. The DSU-110 connects the core to an external memory system and the rest of the *System on Chip* (SoC).

The following figure shows the Neoverse™ N2 core components.

Figure 3-1: Neoverse™ N2 core components



L1 instruction memory system

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

The L1 instruction memory system includes:

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

Instruction decode

The instruction decode unit decodes AArch32 and AArch64 instructions into internal format.

Register rename

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

Instruction issue

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

Integer execute

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

Vector execute

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

Advanced SIMD and floating-point support

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

Cryptographic Extension

The Cryptographic Extension is optional in the Neoverse™ N2 core. The Cryptographic Extension adds new instructions to the Advanced SIMD and the *Scalable Vector Extension* (SVE) instruction sets that accelerate:

- *Advanced Encryption Standard* (AES) encryption and decryption.
- The *Secure Hash Algorithm* (SHA) functions SHA-1, 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 Neoverse™ N2 core license.

Scalable Vector Extension

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

SVE is defined for the AArch64 Execution state only.

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

Note

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

L1 data memory system

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

The L1 data memory system includes:

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

Memory Management Unit

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

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

L2 memory system

The L2 memory system includes the L2 cache. The L2 cache is private to the core and is 8-way set associative. You can configure its RAM size to be 512KB or 1024KB. The L2 memory system is connected to the DSU-110 through an asynchronous CPU bridge.

Embedded Trace Macrocell and Trace Buffer Extension

The Neoverse™ N2 core supports a range of debug, test, and trace options including an *Embedded Trace Macrocell* (ETM) and trace buffer.

The Neoverse™ N2 core also includes a ROM table that contains a list of components in the system. Debuggers can use the ROM table to determine which CoreSight components are implemented.

All the debug and trace components of the Neoverse™ N2 core are described in this manual. The *Arm® Neoverse™ N2 Core Configuration and Integration Manual* provides information about the *Embedded Logic Analyzer* (ELA).

Performance Monitoring Unit

The *Performance Monitoring Unit* (PMU) provides six 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.

Statistical Profiling Extension

The Neoverse™ N2 core implements the optional *Statistical Profiling Extension* (SPE) to the Arm®v8.4-A architecture. The SPE provides a statistical view of the performance characteristics of executed instructions that software writers can use to optimize their code for better performance.

GIC CPU interface

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

CPU bridge

In a cluster, there is one CPU bridge between each Neoverse™ N2 core and the DSU-110.

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

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

Related information

6 Memory management on page 48
7 L1 instruction memory system on page 55
8 L1 data memory system on page 59
9 L2 memory system on page 64
12 GIC CPU interface on page 90
18 Performance Monitors Extension support on page 105
19 Embedded Trace Extension support on page 118
22 Statistical Profiling Extension support on page 132

3.2 Interfaces

The DSU-110 manages all Neoverse™ N2 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 Neoverse™ N2 core implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Armv8-A architectures up to Arm®v8.6-A. The Neoverse™ N2 core supports the AArch64 Execution state at all Exception levels, EL0 to EL3. The Arm®v9.0-A architecture extends the architecture defined in the Armv8-A architectures up to Arm®v8.5-A

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about the programmers model.

Related information

2.4 Supported standards and specifications on page 26

4 Clocks and resets

To provide dynamic power savings, the Neoverse™ N2 core supports hierarchical clock gating. It also supports Warm and Cold resets.

Each Neoverse™ N2 core has a single clock domain and receives a single clock input. This clock input is gated by an architectural clock gate in the CPU bridge.

In addition, the Neoverse™ N2 core implements extensive clock gating that includes:

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

The Neoverse™ N2 core receives the following reset signals from the DSU-110 side of the CPU bridge:

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

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

5 Power management

The Neoverse™ N2 core provides mechanisms to control both dynamic and static power dissipation.

The dynamic power management includes the following features:

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

The static power management includes the following features:

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

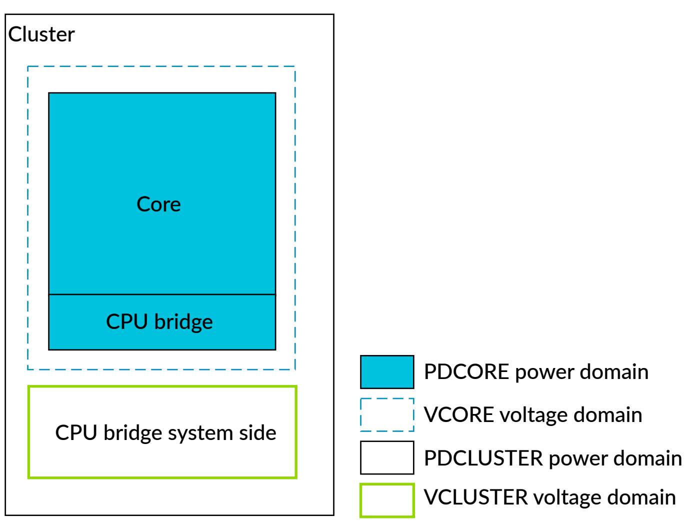
5.1 Voltage and power domains

The DSU-110 *Power Policy Units* (PPUs) control power management for the Neoverse™ N2 core. The core supports one power domain, PDCORE, and one system power domain, PDCLUSTER. Similarly, it supports one core voltage domain, VCORE, and one cluster system voltage domain, VCLUSTER. The power and voltage domains have the same boundaries.

The PDCORE power domain contains all Neoverse™ N2 core logic and part of the core asynchronous bridge that belongs to the VCORE domain. The PDCLUSTER power domain contains the part of the CPU bridge that belongs to the VCLUSTER domain.

The following figure shows the Neoverse™ N2 core power domain and voltage domain. It also shows the cluster power domain and voltage domain that cover the system side of the CPU bridge.

Figure 5-1: Neoverse™ N2 voltage and power domains

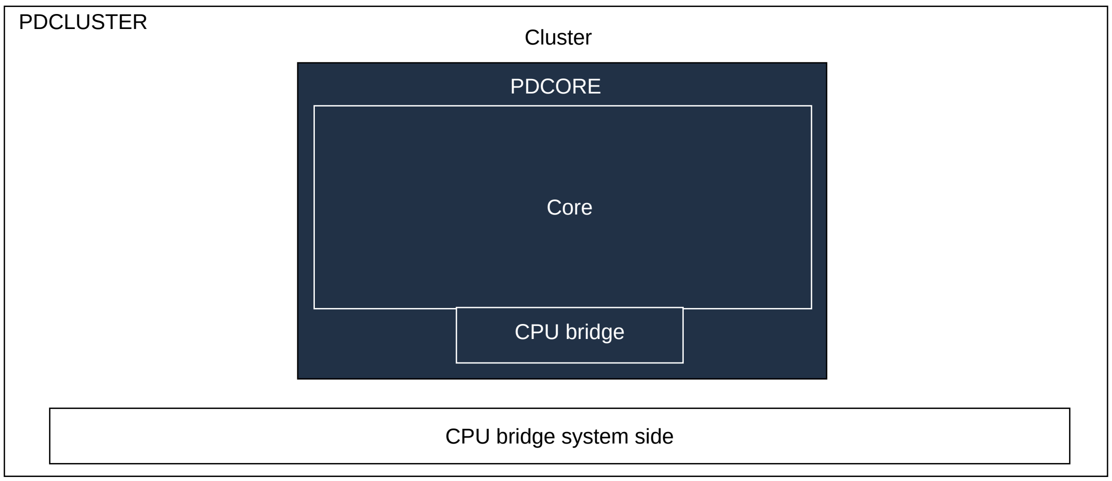


You can tie the VCORE and VCLUSTER voltage domains to the same supply if either:

- 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).

The following figure shows the power domains for an example with a Neoverse™ N2 core in a cluster.

Figure 5-2: Core power domains in a cluster with one Neoverse™ N2 core



Clamping cells between power domains are inferred through power intent files rather than instantiated in the RTL. See *Power management* in the *Arm® Neoverse™ N2 Core Configuration and Integration Manual* for more information.

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*.

Note

The *Arm® Neoverse™ N2 Core Configuration and Integration Manual* is a confidential document that is available with the appropriate product licenses.

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 Armv8, for Armv8-A architecture profile* for more information about entering low-power state and wakeup events.

5.2.2 Low-power state behavior considerations

You must consider the implications of *Wait for Interrupt* (WFI) and *Wait for Event* (WFE) low-power state behavior applicable to the Neoverse™ N2 core.

While the core is in WFI or WFE state, the clocks in the core are temporarily enabled, without causing the core to exit WFI or WFE, 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

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

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about WFI and WFE.

5.3 Power control

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

The core has its own individual PPU for controlling core power domain.

In addition, there is a PPU for the cluster.

The PPUs decide and request any change in power mode. The Neoverse™ N2 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 accepting the request.

See *Power management* and *Power and reset control with Power Policy Units* in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for more information about the PPUs for the cluster and the core.

5.4 Core power modes

The Neoverse™ N2 core power domain has a defined set of power modes and corresponding legal transitions between these modes.

The *Power Policy Unit* (PPU) of a core manages 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 Neoverse™ N2 core power modes.

Caution

Power modes that are not shown in the following table are not supported and must not occur. Deviating from the legal power modes can lead to **UNPREDICTABLE** results. You must comply with the dynamic power management and powerup and powerdown sequences described in 5.5 Neoverse N2 core powerup and powerdown sequence on page 46.

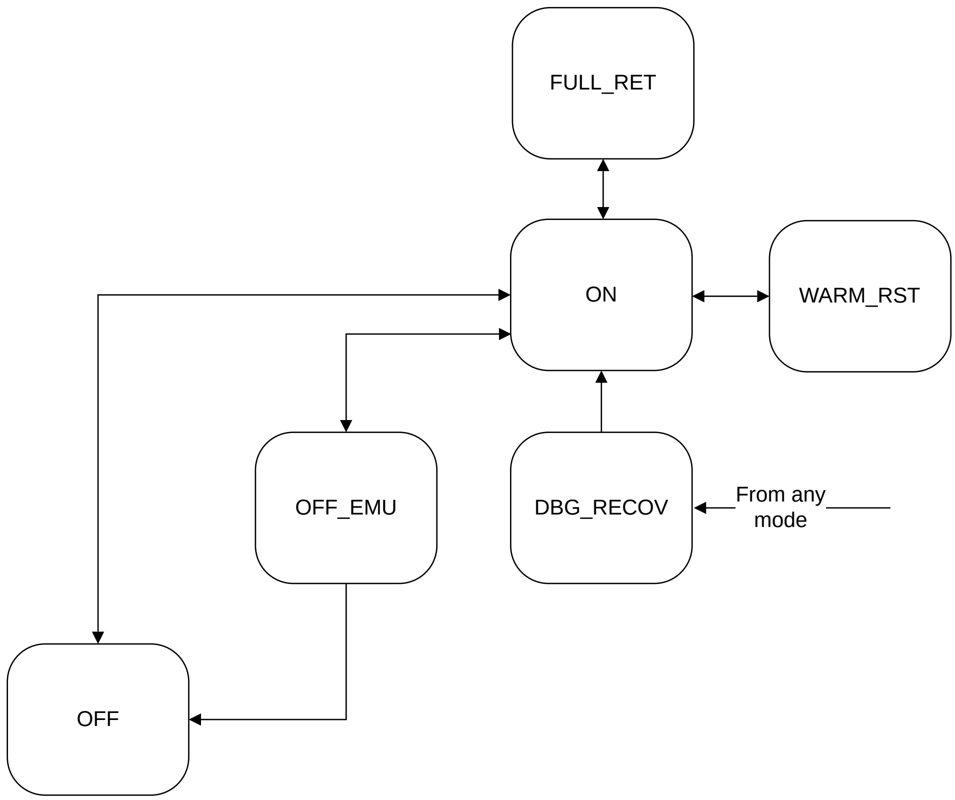
Table 5-1: Neoverse™ N2 core power modes

Power mode	Short name	Power state
On	ON	The core is powered up and active.

Power mode	Short name	Power state
Full retention	FULL_RET	The core is in retention. In this mode, only power that is required to retain register and RAM state is available. The core is not operational. A core must be in *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) low-power state before it enters this mode.
Off	OFF	The core is powered down.
Emulated Off	OFF_EMU	Emulated off mode permits you to debug the powerup and powerdown cycle without changing the software. In this mode, the core 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 Neoverse™ N2 core power domain and the legal transitions between them.

Figure 5-3: Neoverse™ N2 core power mode transitions



Related information

5.2 Architectural clock gating modes on page 40
5.4.4 Full retention mode on page 45
5.2.1 Wait for Interrupt and Wait for Event on page 40

5.4.1 On mode

In the On power mode, the Neoverse™ N2 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 debug access when the core domain is off returns an error response on the internal debug interface, indicating that the core is not available.

5.4.3 Emulated off mode

In Emulated off mode, all core domain logic and RAMs are kept on. All Debug registers must retain their state and be accessible from the external debug interface. All other functional interfaces behave as if the core were 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 B.1.18 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 183.
- 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 Armv8, for Armv8-A architecture profile*.
- An event that requires the core clock to be temporarily enabled without exiting the dynamic retention mode. For example, an L1 or L2 snoop, a cache or TLB maintenance operation, a debug access on the debug APB bus, or a GIC access.

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 DynamIQ™-110 cluster-wide Warm reset must be applied externally. The RAS and cache state are preserved when the core is transitioned to the On mode.

Caution

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 and the debug and *Reliability, Availability, and Serviceability* (RAS) registers.

A Warm reset is applied to the Neoverse™ N2 core when the core receives a Warm reset signal from the DSU-110 side of the CPU bridge:

The Neoverse™ N2 core implements the Arm®v8-A Reset Management Register, RMR_EL3. When running in EL3, setting the RMR_EL3.RR bit to 1 requests a Warm reset.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about RMR_EL3.

5.5 Neoverse™ N2 core powerup and powerdown sequence

No particular sequence applies to the Neoverse™ N2 core powerup. There are no software steps required to bring a core into coherence after reset. For powerdown, the Neoverse™ N2 core uses a specific sequence.

To powerdown the Neoverse™ N2 core:

1. Save all architectural state.
2. Configure the *Generic Interrupt Controller* (GIC) distributor to disable or reroute interrupts away from the core.
3. Set the IMP_CPUPWRCTLR_EL1.CORE_PWRDN_EN bit to 1 to indicate to the power controller that a powerdown is requested.
4. Execute an `ISB` instruction.
5. Execute a `WFI` instruction.

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

- Disables and cleans the L1 cache
- Removes the core from coherency

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

Related information

B.1.18 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 183

5.6 Debug over powerdown

The Neoverse™ N2 core supports debug over powerdown, which allows a debugger to retain its connection with the core even when powered down. This behavior enables debug to continue through powerdown scenarios, rather than having to re-establish a connection each time the core is powered up.

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

See *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 Neoverse™ N2 core internal registers and translation tables. The MMU also controls memory access permissions, memory ordering, and cache policies for each region of memory.

An address translation from an input address to an output address is described as a stage of address translation. The Neoverse™ N2 core can perform:

- Stage 1 translations that translate an input *Virtual Address* (VA) to an output *Physical Address* (PA) or *Intermediate Physical Address* (IPA).
- Stage 2 translations that translate an input IPA to an output PA.
- Combined stage 1 and stage 2 translations that translate an input VA to an IPA, and then translate that IPA to an output PA. The Neoverse™ N2 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 Armv8, for Armv8-A architecture profile* for more information on this feature.

6.1 Memory Management Unit components

The Neoverse™ N2 *Memory Management Unit* (MMU) includes several *Translation Lookaside Buffers* (TLBs), an *MMU Translation Cache* (MMUTC), and a translation table prefetcher.

A TLB is a cache of recently executed page translations within the MMU. The Neoverse™ N2 core implements a two-level TLB structure. The TLB stores all translation table sizes and is responsible for breaking these down into smaller tables when required for the L1 data or instruction TLB.

The following table describes the MMU components.

Table 6-1: MMU components

Component	Description
L1 instruction TLB	• Caches entries at the 4KB, 16KB, 64KB, or 2MB granularity of *Virtual Address* (VA) to *Physical Address* (PA) mapping only • Fully associative • 48 entries
L1 data TLB	• Caches entries at the 4KB, 16KB, 64KB, 2MB, or 512MB granularity of VA to PA mappings only • Fully associative • 44 entries
L1 *Trace Buffer Extension* (TRBE) TLB	• VA to PA translations of any page and block size • 2 entries

L2 TLB	• Shared by instructions and data
	• VA to PA mappings for 4KB, 16KB, 64KB, 2MB, 32MB, 512MB, and 1GB block sizes
	• *Intermediate Physical Address* (IPA) to PA mappings for:
	  ◦ 2MB and 1GB block sizes in a 4KB translation granule
	  ◦ 32MB block size in a 16KB translation granule
	  ◦ 512MB block size in a 64KB granule
	• *Intermediate PAs* (IPAs) obtained during a translation table walk
	• 5-way set associative
	• 1280 entries

Translation table prefetcher	• Detects access to contiguous translation tables and prefetches the next one • Can be disabled in the ECTLR register

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

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

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

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

A miss in the L1 data TLB or a hit in the L2 TLB has a 3-cycle penalty compared to a hit in the L1 data TLB. This penalty can be increased depending on the arbitration of pending requests.

6.2 Translation Lookaside Buffer entry content

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

Each TLB entry contains:

- A *Virtual Address* (VA)
- A *Physical Address* (PA)
- 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 51

6.3 Translation Lookaside Buffer match process

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

Each TLB entry is associated with a particular translation regime.

- EL3 in Secure state
- EL2 (or EL0 in VHE mode) in Secure state
- EL1 or EL0 in Secure state
- EL1 or EL0 in Non-secure state

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

- The VA bits[48:N], where N is $\log_2$ of the block size for that translation that is stored in the TLB entry, matches the requested address.
- Entry translation regime matches the current translation regime.
- The 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 VMID matches the current VMID held in the VTTBR_EL2 register.

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

- EL2 in Secure state with HCR_EL2.E2H and HCR_EL2.TGE set to 1
- EL1 or EL0 in Secure state
- EL1 or EL0 in Non-secure state

VMID is relevant for EL1 or EL0 in Non-secure state when HCR_EL2.E2H and HCR_EL2.TGE are not both set. It is also relevant in Secure state when SCR_EL3.EEL2 is 1.

6.4 Translation table walks

When the Neoverse™ N2 core generates a memory access, the *Memory Management Unit* (MMU) searches for the requested *Virtual Address* (VA) in the *Translation Lookaside Buffers* (TLBs). If it is not present, then it is a miss and the MMU proceeds by looking up the translation table during a translation table walk.

When the Neoverse™ N2 core generates a memory access, the MMU:

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

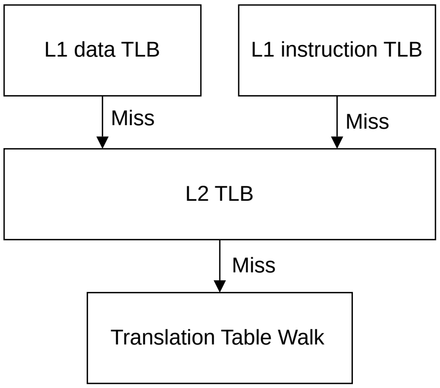
Address translation is performed only when the MMU is enabled. They can also be disabled for a particular translation base register, in which case the MMU returns a translation fault.

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

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

The following figure shows the translation table walk process.

Figure 6-1: Translation table walks



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

Related information

7 L1 instruction memory system on page 55
8 L1 data memory system on page 59
9 L2 memory system on page 64

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 Neoverse™ N2 core supports hardware updates to the Access flag and to dirty state only when the translation tables are held in Inner Write-Back and Outer Write-Back Normal memory regions. If software requests a hardware update in a region that is not Inner Write-Back or Outer Write-Back Normal memory, then the Neoverse™ N2 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, data accesses due to all loads to Normal memory, all loads with acquire semantics and all `AtomicLd`, `AtomicCAS`, and `AtomicSwap` instructions. The address captured in the *Fault Address Register* (FAR) is the target address of the instruction that generated the synchronous abort. External aborts are reported asynchronously, then they occur for loads to Device memory without release semantics, stores to any memory type, and `AtomicSt`, cache maintenance, `TLBI`, and `IC` instructions.

Misprogramming contiguous hints

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

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

If there is this kind of mis-programming, the Neoverse™ N2 core does not generate a translation fault.

Conflict aborts

Conflict aborts are generated from the L1 TLB. If a conflict abort is detected in the L2 TLB, then it chooses one valid translation and a conflict abort is not generated.

6.7 Memory behavior and supported memory types

The Neoverse™ N2 core supports memory types defined in the Armv8-A architecture.

The following table shows how memory types are supported in the Neoverse™ N2 core.

Table 6-2: Supported device memory types

Memory attribute type	Shareability	Inner Cacheability	Outer Cacheability	Notes
Device nGnRnE	Outer Shareable	-	-	Treated as Device nGnRnE
Device nGnRE	Outer Shareable[1]	-	-	Treated as Device nGnRE
Device nGRE	Outer Shareable[1]	-	-	Treated as Device nGRE
Device GRE	Outer Shareable[1]	-	-	Treated as Device GRE
Normal	Outer Shareable[1]	Non-cacheable	Any	Treated as Non-cacheable
Normal	Outer Shareable[1]	Write-Through Cacheable	Any	Treated as Non-cacheable
Normal	Outer Shareable[1]	Write-Back Cacheable	Non-cacheable	Treated as Non-cacheable
Normal	Outer Shareable[1]	Write-Back Cacheable	Write-Through Cacheable	Treated as Non-cacheable
Normal	See Table 6-3: Shareability for Normal memory on page 54.	Write-Back Cacheable (any allocation hint)	Write-Back Cacheable No Allocate	Treated as Write-Back Read and Write Allocate but the outer cacheability propagated to the DSU-110 is 0 (No Allocate)
Normal	See Table 6-3: Shareability for Normal memory on page 54.	Write-Back Cacheable (any allocation hint)	Write-Back Read or Write Allocate	Treated as Write-Back Read and Write Allocate but the outer cacheability propagated to the DSU-110 is 1, therefore upgraded to Write and Read Allocate

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

Table 6-3: Shareability for Normal memory

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

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

7 L1 instruction memory system

The Neoverse™ N2 L1 memory system is responsible for fetching instructions and predicting branches. It includes the L1 instruction cache, the L1 instruction *Translation Lookaside Buffer* (TLB), and the *Macro-operation* (MOP) cache.

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

The following table shows the L1 instruction memory system features.

Table 7-1: L1 instruction memory system features

Feature	Description
L1 instruction cache	64KB
	4-way set associative
	Virtually Indexed, Physically Tagged (VIPT) behaving as *Physically Indexed, Physically Tagged* (PIPT)
	Always protected with parity
Cache line length	64 bytes
Macro-operation (MOP) cache	1536 Macro-operations
	4-way skewed associative
	Virtually Indexed, Virtually Tagged (VIVT) behaving as *Physically Indexed, Physically Tagged* (PIPT)
	Level 0 instruction cache working in the fetch stages of the pipeline to improve throughput and latency
Cache policy	Pseudo-*Least Recently Used* (LRU) cache replacement policy

Note

The L1 instruction TLB also resides in the L1 instruction memory system. However, it is part of the *Memory Management Unit* (MMU) and is described in 6 Memory management on page 48.

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

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

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

Note

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

Related information

5.4.5 Debug recovery mode on page 45

7.2 L1 instruction cache Speculative memory accesses

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

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

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

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

7.3 Program flow prediction

The Neoverse™ N2 core contains program flow prediction hardware, also known as branch prediction. Branch prediction increases overall performance and reduces power consumption.

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

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

Program flow prediction hardware contains the following functionality:

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

Predicted and non-predicted instructions

Unless otherwise specified, the following list applies to A64, A32, and T32 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
- Branches that switch between A32 and T32 states

Exception return branch instructions are not predicted.

T32 state conditional branches

A T32 unconditional branch instruction can be made conditional by inclusion in an *If-Then* (IT) block. It is then treated as a conditional branch.

Return stack

The return stack stores the address and instruction set state. This address is equal to the link register value stored in R14 in AArch32 state or X30 in AArch64 state.

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`

In AArch32, any of the following instructions causes a return stack push:

- `BL r14`
- `BLX`

- `MOV pc, r14`

Any of the following instructions cause a return stack pop:

- `BX`
- `LDR pc, [r13], #imm`
- `LDM r13, {...pc}`

The following instructions are not predicted:

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

7.4 Instruction cache hardware coherency

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

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

The following restrictions and recommendations apply to configuring instruction cache hardware coherency in the core:

- The coherency domain containing a core configured with instruction cache hardware coherency must not contain any coherent masters that require software instruction cache maintenance.
- Arm recommends that systems using instruction cache hardware coherency should be configured with an L2 cache size of 1MB. An L2 cache size of 512KB is also acceptable, but will see approximately a 1-2% reduction in performance due to the overhead of a strictly inclusive L1 instruction cache and L2 cache.
- Arm recommends systems consisting of a large number of Neoverse™ N2 cores should configure the cores with instruction cache coherency to eliminate possible performance issues related to instruction cache instruction broadcasts as DVMMsg transactions to all masters in the system.

8 L1 data memory system

The Neoverse™ N2 L1 data memory system is responsible for executing load and store instructions, as well as specific instructions such as atomics, cache maintenance operations, and memory tagging instructions. It includes the L1 data cache and the L1 data *Translation Lookaside Buffer* (TLB).

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

The following table shows the L1 data memory system features.

Table 8-1: L1 data memory system features

Feature	Description
L1 data cache	64KB
	4-way set associative
	Virtually indexed, Physically Tagged (VIPT) behaving as *Physically Indexed, Physically Tagged* (PIPT)
	Always protected with *Error Correcting Code* (ECC)
Cache line length	64 bytes
Cache policy	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 • 3×128-bit read paths and 2×128-bit write paths for the vector execute pipeline

Note

The L1 data TLB also resides in the L1 data memory system. However, it is part of the *Memory Management Unit* (MMU) and is described in 6 Memory management on page 48.

8.1 L1 data cache behavior

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

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

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

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 an instruction fetch.

- All load and store instructions to cacheable memory are treated as Non-cacheable.
- Data cache maintenance operations continue to execute normally.

The L1 data and L2 caches cannot be disabled independently. When a core disables the L1 data cache, cacheable memory accesses issued by that core are no longer cached in the L1 or L2 cache. 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 Neoverse™ N2 core uses the *Modified Exclusive Shared Invalid* (MESI) protocol.

Related information

5.4.5 Debug recovery mode on page 45

8.2 Instruction implementation in the L1 data memory system

The Neoverse™ N2 core supports the atomic instructions added in the Arm®v8.1-A architecture. Atomic instructions to Cacheable memory can be performed as either near atomics or far atomics, depending on where the cache line containing the data resides.

If an instruction hits in the L1 data cache, then the Neoverse™ N2 core tries to perform it as a near atomic. Then, based on system behavior, the core can decide to perform it as a far atomic.

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

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

The Neoverse™ N2 core supports atomics to Device or Non-cacheable memory, however this relies on the interconnect also supporting atomics. If such an atomic instruction is executed when the interconnect does not support them, then it results in an abort.

8.3 Internal exclusive monitor

The Neoverse™ N2 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/store exclusive instruction is any of the following:

- In the A32 and T32 instruction sets, any instruction that has a mnemonic starting with `LDREX`, `STREX`, `LDAEX`, or `STLEX`.
- In the A64 instruction set, any instruction that has a mnemonic starting with `LDX`, `LDAX`, `STX`, or `STLX`.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information on these instructions.

8.4 Data prefetching

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

Preload instructions

The Neoverse™ N2 core supports the AArch64 prefetch memory instructions, `PRFM`, the AArch32 Prefetch Data, `PLD`, and the AArch32 Preload Data With Intent To Write, `PLDW`, instructions.

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

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

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

For more information about prefetch memory and preloading caches, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Data prefetching and monitoring

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

The CPUECTLR register allows you to have some control over the prefetcher.

Data cache zero

In the Neoverse™ N2 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 Armv8, for Armv8-A architecture profile*.

8.5 Write streaming mode

The Neoverse™ N2 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 was subsequently written by the `memset()`. In some situations, cache line allocation on writes is not required. For example, when executing the C standard library `memset()` function to clear a large block of memory to a known value.

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

When in write streaming mode, load operations behave as normal, and can still cause linefills. Writes still lookup in the cache, but if they miss then they write out to the L2 or L3 cache rather than starting a linefill.

Note

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

The BIU continues in write streaming mode until either:

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

When a Neoverse™ N2 core has switched to write streaming mode, the BIU 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:

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

Related information

B.1.15 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 166

9 L2 memory system

The Neoverse™ N2 L2 memory system connects the core with the DSU-110 through the CPU bridge. It includes the L2 *Translation Lookaside Buffer* (TLB) and private L2 cache.

The L2 cache is unified and private to each Neoverse™ N2 core in a cluster.

Note

For some cores, you can implement the DSU-110 to use the Direct connect feature to connect to the core. However, the Neoverse™ N2 core supports Direct connect only.

The following table shows the L2 memory system features.

Table 9-1: L2 memory system features

Feature	Type
L2 cache	512KB or 1024KB
	8-way set associative, 4 banks, 2 banks
	Physically-Indexed, Physically-Tagged (PIPT)
	Always protected with *Error Correcting Code* (ECC)
Cache line length	64 bytes
Cache policy	Dynamic biased cache replacement policy
Interface with the DSU-110	One CHI Issue E compliant interface with 256-bit read and write DAT channel widths

9.1 L2 cache

The integrated L2 cache handles both instruction and data requests from the instruction and data side, 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 inclusive. Any data contained in the L1 data cache is also present in the L2 cache. Victimization of L2 data can cause invalidations of the L1 data cache.

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

Related information

5.4.5 Debug recovery mode on page 45

9.2 Support for memory types

The Neoverse™ N2 core simplifies the 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 allocate into the L1 cache and thus the L2 cache due to inclusivity.
An allocating read to the L1 data cache that has the transient bit set is allocated in the L1 cache. Such reads are marked as most likely to be evicted, according to the L1 eviction policy.Transient lines evicted from the L2 cache do not allocate downstream caches.

9.3 Transaction capabilities

The CHI Issue E interface between the Neoverse™ N2 L2 memory system and the DSU-110 provides transaction capabilities for the core.

The following table shows the maximum possible values for read, write, *Distributed Virtual Memory* (DVM) issuing, and snoop capabilities of the Neoverse™ N2 L2 cache.

Table 9-2: Neoverse™ N2 transaction capabilities

Attribute	Maximum value	Description
Write issuing capability	46, 54, or 62	This is the maximum number of outstanding write transactions. It depends on the configured *Transaction Queue* (TQ) size: 46, 54, or 62.
Read issuing capability	46, 54, or 62	This is the maximum number of outstanding read transactions. It depends on the configured TQ size: 46, 54, or 62.
Snoop acceptance capability	29, 33, or 37	This is the maximum number of outstanding snoops accepted. It depends on the configured TQ size: 29, 33, or 37.
DVM issuing capability	46, 54, or 62	This is the maximum number of outstanding DVM operation transactions. It depends on the configured TQ size: 46, 54, or 62.

10 Direct access to internal memory

The Neoverse™ N2 core provides a mechanism to read the internal memory that the L1 and L2 caches and TLB structures use, through **IMPLEMENTATION DEFINED** system registers. This functionality can be useful when investigating issues where the coherency between the data in the cache and data in system memory is broken.

Note

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

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

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

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

10.1 L1 cache encodings

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

The size of the configured cache determines the number of sets in each way. The encoding that is used to locate the cache data entry for tag and data memory is set in `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: Neoverse™ N2 L1 instruction cache tag location encoding

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

Bit field of Xn	Description
[13:6]	Virtual address [13:6]
[5:0]	Reserved

Table 10-3: Neoverse™ N2 L1 instruction cache data location encoding

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

Table 10-4: Neoverse™ N2 L1 BTB data location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x02`
[23:16]	Reserved
[15:4]	Index [11:0]
[3:0]	Reserved

Table 10-5: Neoverse™ N2 L1 GHB data location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x03`
[23:15]	Reserved
[14:5]	Index [9:0]
[4:0]	Reserved

Table 10-6: Neoverse™ N2 L1 instruction TLB data location encoding

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

Table 10-7: Neoverse™ N2 BIM data location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x05`
[23:14]	Reserved
[13:4]	Index [9:0]
[3:0]	Reserved

Table 10-8: Neoverse™ N2 L0 Macro-operation cache data location encoding

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

Table 10-9: Neoverse™ N2 L1 data cache tag location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x08`
[23:20]	Reserved
[19:18]	Way
[17:16]	Copy: **`0b00`** Tag RAM associated with Pipe 0 **`0b01`** Tag RAM associated with Pipe 1 **`0b10`** Tag RAM associated with Pipe 2 **`0b11`** Reserved
[15:14]	Reserved
[13:6]	Physical address [13:6]
[5:0]	Reserved

Table 10-10: Neoverse™ N2 L1 data cache data location encoding

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

Table 10-11: Neoverse™ N2 L1 data TLB location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x0A`
[23:6]	Reserved
[5:0]	TLB Entry (0-43)

10.1.1 L1 instruction tag RAM returned data

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

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

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

Bit field	Description
[39]	Non-secure identifier for the physical address
[38:3]	Physical address [47:12]
[2:1]	Instruction state [1:0] **`0b00`** Invalid **`0b01`** T32 **`0b10`** A32 **`0b11`** A64
[0]	Parity

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

Bit field	Description
[63:0]	0

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

Bit field	Description
[63:0]	0

10.1.2 L1 instruction data RAM returned data

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

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

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

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

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

Bit field	Description
[63:9]	0

Bit field	Description
[8:0]	Data [72:64]

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

Bit field	Description
[63:0]	0

10.1.3 L1 BTB RAM returned data

For each register, any access to the L1 *Branch Target Buffer* (BTB) RAM returns data.

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

Table 10-18: L1 BTB cache format for Instruction Register 0

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

Table 10-19: L1 BTB cache format for Instruction Register 1

Bit field	Description
[63:28]	0
[27:0]	Data [91:64]

Table 10-20: L1 BTB cache format for Instruction Register 2

Bit field	Description
[63:0]	0

10.1.4 L1 GHB RAM returned data

For each register, any access to the L1 *Global History Buffer* (GHB) RAM returns data.

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

Table 10-21: L1 GHB cache format for Instruction Register 0

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

Table 10-22: L1 GHB cache format for Instruction Register 1

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

Table 10-23: L1 GHB cache format for Instruction Register 2

Bit field	Description
[63:0]	0

10.1.5 L1 BIM RAM returned data

For each register, any access to the L1 *Bimodal Predictor* (BIM) RAM returns data.

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

Table 10-24: L1 BIM cache format for Instruction Register 0

Bit field	Description
[63:16]	0
[15:0]	Data [15:0]

Table 10-25: L1 BIM cache format for Instruction Register 1

Bit field	Description
[63:0]	0

Table 10-26: L1 BIM cache format for Instruction Register 2

Bit field	Description
[63:0]	0

10.1.6 L1 instruction TLB returned data

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

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

Table 10-27: L1 instruction TLB format for Instruction Register 0

Bit field	Description
[63:61]	Virtual address [14:12]
[60:59]	PBHA [1:0]
[58]	TLB attribute

Bit field	Description

[57:55]	Memory attributes:
	`0b000`
	Device nGnRnE
	`0b001`
	Device nGnRE
	`0b010`
	Device nGRE
	`0b011`
	Device GRE
	`0b100`
	Non-cacheable
	`0b101`
	Write-Back No-Allocate
	`0b110`
	Write-Back Transient
	`0b111`
	Write-Back Read-Allocate and Write-Allocate

[54:52]	Page size:
	`0b000`
	4KB
	`0b001`
	16KB
	`0b010`
	64KB
	`0b100`
	2MB
	Other
	Reserved

[51:48]	TLB attribute
[47]	Outer-shared
[46]	Inner-shared
[45:40]	TLB attribute
[39:24]	ASID[15:0]
[23:8]	VMID[15:0]

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

Table 10-28: L1 instruction TLB format for Instruction Register 1

Bit field	Description
[63]	TLB attribute
[62]	Non-secure
[63:34]	Physical address [41:12]
[33:0]	Virtual address [48:15]

Table 10-29: L1 instruction TLB format for Instruction Register 2

Bit field	Description
[63:9]	Reserved
[8:7]	TLB attribute
[6]	Non-secure
[5:0]	Physical address [47:42]

10.1.7 L0 macro-operation RAM returned data

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

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

Table 10-30: L0 MOP cache format for Instruction Register 0

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

Table 10-31: L0 MOP cache format for Instruction Register 1

Bit field	Description
[63:40]	0

Bit field	Description
[39:0]	Macro-operation data [103:64]

Table 10-32: L0 MOP cache format for Instruction Register 2

Bit field	Description
[63:0]	0

10.1.8 L1 data tag RAM returned data

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

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

Table 10-33: L1 data cache tag format for Data Register 0

Bit field	Description
[63:27]	Non-secure identifier, physical address [47:12]
[26]	Origin
[25]	Prefetch
[24]	Transient/WBNA
[23:20]	*Memory Tagging Extension* (MTE) tag poison
[19:4]	MTE tag data
[3:2]	MTE tag state: **`0b00`** Invalid **`0b01`** Shared **`0b11`** Dirty
[1:0]	*Modified Exclusive Shared Invalid* (MESI): **`0b00`** Invalid **`0b01`** Shared **`0b10`** Exclusive **`0b11`** Modified

Table 10-34: L1 data cache tag format for Data Register 1

Bit field	Description
[63:8]	0
[7:0]	ECC

Table 10-35: L1 data cache tag format for Data Register 2

Bit field	Description
[63:0]	0

10.1.9 L1 data data RAM returned data

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

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

Table 10-36: L1 data cache data format for Data Register 0

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

Table 10-37: L1 data cache data format for Data Register 1

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

Table 10-38: L1 data cache data format for Data Register 2

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

10.1.10 L1 data TLB returned data

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

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

Table 10-39: L1 data TLB format for Data Register 0

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

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

Bit field	Description
[36]	Non-secure
[35:33]	MSID [2:0]
[32:17]	ASID [15:0]
[16:1]	VMID [15:0]
[0]	Valid

Table 10-40: L1 data TLB format for Data Register 1

Bit field	Description
[63:35]	Physical address [40:12]
[34:0]	Virtual address [48:14]

Table 10-41: L1 data TLB format for Data Register 2

Bit field	Description
[63:14]	Reserved
[13]	Nested virtualization
[12]	Tagged MTE
[11]	Forced Write-Back override
[10:7]	PBHA [3:0]
[6:0]	PA [47:41]

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 10-42: Neoverse™ N2 L2 cache tag location encoding for 512KB [2]

Bit field of Xn	Description
[31:24]	RAMID = `0x10`
[23:21]	RES0
[20:18]	Way (0-7)
[17:16]	RES0
[15:12]	Index [15:12]
[11:9]	XOR(Index [11:9], Way [2:0])

[2] index[15:7]=XOR(Physical Address[15:7],Physical Address[24:16]) index[6]=XOR(Physical Address[6], Physical Address[10])

Bit field of Xn	Description
[8:7]	Index [8:7]
[6]	XOR(index [6], Index [10], Way [1])
[5:0]	RES0

Table 10-43: Neoverse™ N2 L2 cache tag location encoding for 1MB

Bit field of Xn	Description
[31:24]	RAMID = `0x10`
[23:21]	RES0
[20:18]	Way (0-7)
[17]	RES0
[16:11]	Index [16:11]
[10:8]	XOR(Index [10:8], Way [2:0])
[7]	Index [7]
[6]	XOR(Index [6], Index [10], Way [2])
[5:0]	RES0

Table 10-44: Neoverse™ N2 L2 cache data location encoding for 512KB

Bit field of Xn	Description
[31:24]	RAMID = `0x11`
[23:21]	RES0
[20:18]	Way (0-7)
[17:16]	RES0
[15:12]	XOR(Index [15:12], 4'b1000)
[11:9]	XOR(Index [11:9], Way [2:0])
[8:7]	Index [8:7]
[6]	XOR(Index [6], Index [10], Way [1])
[5:4]	Physical address [5:4]
[3:0]	RES0

Table 10-45: Neoverse™ N2 L2 cache data location encoding for 1MB

Bit field of Xn	Description
[31:24]	RAMID = `0x11`
[23:21]	RES0
[20:18]	Way (0-7)
[17]	RES0
[16:11]	XOR(Index [16:11], 6'b001000)
[10:8]	XOR(Index [10:8], Way [2:0])
[7]	Index [7]
[6]	XOR(Index [6], Index [10], Way [2])
[5:4]	Physical address [5:4]
[3:0]	RES0

Table 10-46: Neoverse™ N2 L2 TLB location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x18`
[23:21]	RES0
[20:18]	Way (0-5)
[17:8]	RES0
[7:0]	TLB entry (0-255)

Table 10-47: Neoverse™ N2 L2 victim location encoding

Bit field of Rd	Description
[31:24]	RAMID = `0x12`
[23:17]	RES0
[16:8]	XOR(Index [16:8], 9'b010000000)
[7:6]	XOR(Index [6],Index [10])
[5:4]	Physical address [5:4]
[3:0]	RES0

10.2.1 L2 tag RAM returned data

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

The following tables show the L2 cache tag format for data registers when configured without coherent instruction cache. In the first table:

For 512KB L2 cache

n=41, m=16

For 1MB L2 cache

n=40, m=17

Table 10-48: L2 tag cache format for Data Register 0

Bit field	Description
[63:n+23]	0
[n+22:n+16	ECC
[n+15]	MPAM_PMG
[n+14:n+6]	MPAM_PARTID
[n+5]	MPAM_NS
[n+4:n+1]	PBHA[3:0]
[n:10]	Physical tag [47:m]
[9]	Non-secure
[8:7]	Virtual address [13:12]
[6]	Shareable
[5]	L1 data cache valid

Bit field	Description
[4:3]	MTE state: **0b00** Invalid **0b10** Clean **0b11** Dirty
[2:0]	L2 state: **0b101** UniqueDirty **0b001** UniqueClean **0bx11** SharedClean **0bxx0** Invalid

Table 10-49: L2 tag cache format for Data Register 1

Bit field	Description
[63:0]	0

Table 10-50: L2 tag cache format for Data Register 2

Bit field	Description
[63:0]	0

The following tables show the L2 tag cache format for data registers when configured with coherent instruction cache:

For 512KB L2 cache
n=42, m=16

For 1MB L2 cache
n=41, m=17

Table 10-51: L2 tag cache format for Data Register 0

Bit field	Description
[63:n+20]	ECC
[n+19]	MPAM_PMG
[n+18:n+10]	MPAM_PARTID
[n+9]	MPAM_NS
[n+8:n+5]	L1 instruction cache valid[3:0]
[n+4:n+1]	PBHA[3:0]
[n:11]	Physical tag [47:m]

Bit field	Description
[10]	Non-secure
[9:8]	Virtual address [13:12]
[7]	Shareable
[6]	L1 data cache shared
[5]	L1 data cache valid
[4:3]	MTE state: **0b00** Invalid **0b10** Clean **0b11** Dirty
[2:0]	L2 state: **0b101** UniqueDirty **0b001** UniqueClean **0bx11** SharedClean **0bxx0** Invalid

Table 10-52: L2 tag cache format for Data Register 1

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

Table 10-53: L2 tag cache format for Data Register 2

Bit field	Description
[63:0]	0

10.2.2 L2 data RAM returned data

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

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

Table 10-54: L2 data RAM format for Data Register 0

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

Table 10-55: L2 data RAM format for Data Register 1

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

Table 10-56: L2 data RAM format for Data Register 2

Bit field	Description
[63:20]	0
[19:4]	[15:8] is ECC for Data [127:0], [7:0] is ECC for Data [63:0]
[3:0]	MTE tags

10.2.3 L2 TLB RAM returned data

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

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

Table 10-57: L2 TLB format for Instruction Register 0

Bit field	Description
[63:62]	Reserved
[61:26]	Physical address [47:12]
[25:20]	Reserved
[19:17]	Page size: **`0b000`** 4KB **`0b001`** 16KB **`0b010`** 64KB **`0b100`** 2MB **`0b101`** 32MB **`0b110`** 512MB **`0b111`** 1GB
[16:7]	Reserved
[6]	Coalesced entry
[5:2]	Valid bits
[1:0]	Reserved

Table 10-58: L2 TLB format for Instruction Register 1

Bit field	Description
[63:59]	ASID [3:0]
[58:55]	PBHA
[54]	Walk cache entry
[53:25]	Virtual address [48:20]
[24:21]	Reserved
[20]	Non-secure
[19:9]	Reserved
[8]	nG, indicates a non global page
[7]	Outer shareable
[6]	Inner shareable
[5]	Outer allocate
[4:2]	Memory attributes: **`0b000`** Device nGnRnE **`0b001`** Device nGnRE **`0b010`** Device nGRE **`0b011`** Device GRE **`0b100`** Non-cacheable **`0b101`** Write-Back No-Allocate **`0b110`** Write-Back Transient **`0b111`** Write-Back Read-Allocate and Write-Allocate
[1:0]	Reserved

Table 10-59: L2 TLB format for Instruction Register 2

Bit field	Description
[63:31]	Reserved

Bit field	Description
[30:28]	MSID [2:0]: **0b000** Secure EL1 **0b001** Secure EL2 **0b010** Non-secure EL1 **0b011** Non-secure EL2 **0b101** EL3
[27:12]	VMID [15:0]
[11:0]	ASID [15:4]

10.2.4 L2 Victim RAM returned data

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

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

Table 10-60: Neoverse™ N2 L2 victim format for data register 0

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

Table 10-61: Neoverse™ N2 L2 victim format for data register 1

Bit field of Rd	Description
[63:0]	0

Table 10-62: Neoverse™ N2 L2 victim format for data register 2

Bit field of Rd	Description
[63:0]	0

11 RAS extension support

The Neoverse™ N2 core supports the *Reliability, Availability, and Serviceability* (RAS) Extension, including all extensions up to Arm®v9.0-A.

In particular, the Neoverse™ N2 core supports:

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

The Neoverse™ N2 core features the following nodes:

- Node 0 that includes the shared L3 memory system in the 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 Neoverse™ N2 core includes cache protection. In this case, the Neoverse™ N2 core protects against errors that result in a RAM bitcell holding the incorrect value.

The RAMs in the Neoverse™ N2 core have the following capability:

SED parity

Single Error Detect. 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. When the datum and code bits are all-zero or all-one, the interpretation is that an error has occurred that the ECC scheme cannot correct. However, it might be corrected by other means, such as refetching cached data.

The following table shows which protection type is applied to each RAM in the Neoverse™ N2 core. The core can progress and remain functionally correct when there is a single bit error in any RAM.

Table 11-1: RAM cache protection

RAM	ECC or parity
L1 instruction cache data	SED parity
L1 instruction cache tag	SED parity
L0 *Macro-operation* (MOP) cache data	SED parity
L1 data cache data	SECDED ECC
L1 data cache tag	SECDED ECC
MMU Translation Cache (MMUTC)	SED parity
L2 cache data	SECDED ECC
L2 cache tag	SECDED ECC
L2 *Transaction Queue* (TQ)	SECDED ECC

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

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

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

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

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

11.2 Error containment

The Neoverse™ N2 core supports error containment, which means that an error is detected and not silently propagated.

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

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

11.3 Fault detection and reporting

When the Neoverse™ N2 core detects a fault, it raises a *Fault Handling Interrupt* (FHI) exception or an *Error Recovery Interrupt* (ERI) exception through the fault or the error signals. FHIs and ERIs are reflected in the *Reliability, Availability, and Serviceability* (RAS) registers, which are updated in the node that detects the errors.

Fault handling interrupts

When ERR1CTLR.FI is set, all detected Deferred errors, Uncorrected errors, and overflows of the corrected error counters cause an FHI to be generated. When ERR1CTLR.CFI is set, all detected Corrected errors also cause an FHI to be generated.

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

Error recovery interrupts

When ERR1CTLR.UI is set, all detected Uncorrected errors that are not deferred generate an ERI.

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

Related information

11.6 AArch64 RAS register summary on page 89
B.11.4 ERXCTLR_EL1, Selected Error Record Control Register on page 423
B.11.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register on page 426

11.4 Error detection and reporting

When the Neoverse™ N2 core consumes an error, it raises different exceptions depending on the error type.

The Neoverse™ N2 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, ERR<n>FR. These read-only registers specify various error record settings.
- Error Record Control Registers, ERR<n>CTLR. These registers enable error reporting and also enable various interrupts that are related to errors and faults.
- Error Record Miscellaneous Registers, ERR<n>MISC0-3. These registers record details of the error location and counts.
- Pseudo-fault Generation Feature register, ERR<n>PFGF. 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 Armv8, for Armv8-A architecture profile* for a description of MDCR_EL3.

Related information

18.1 Performance monitors events on page 105

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 Neoverse™ N2 core can inject the following error types:

Corrected errors

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

Deferred errors

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

Uncontainable errors

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

An error can be injected immediately or when a 32-bit counter reaches zero. You can control the value of the counter through the Error Pseudo-fault Generation Countdown Register, ERR0PFGCDN. 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 ERR0PFGCDN.

Note

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

11.6 AArch64 RAS register summary

The summary table provides an overview of all implementation defined ras registers in the core. Individual register descriptions provide detailed information.

Table 11-2: ras register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
ERRIDR_EL1	3	C5	0	C3	0	See individual bit resets.	64-bit	Error Record ID Register
ERRSELR_EL1	3	C5	0	C3	1	See individual bit resets.	64-bit	Error Record Select Register
ERXFR_EL1	3	C5	0	C4	0	See individual bit resets.	64-bit	Selected Error Record Feature Register
ERXCTLR_EL1	3	C5	0	C4	1	0x0	64-bit	Selected Error Record Control Register
ERXSTATUS_EL1	3	C5	0	C4	2	0x0	64-bit	Selected Error Record Primary Status Register
ERXADDR_EL1	3	C5	0	C4	3	See individual bit resets.	64-bit	Selected Error Record Address Register
ERXPFGF_EL1	3	C5	0	C4	4	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Feature register
ERXPFGCTL_EL1	3	C5	0	C4	5	0x0	64-bit	Selected Pseudo-fault Generation Control register
ERXPFGCDN_EL1	3	C5	0	C4	6	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Countdown register
ERXMISC0_EL1	3	C5	0	C5	0	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 0
ERXMISC1_EL1	3	C5	0	C5	1	0x0	64-bit	Selected Error Record Miscellaneous Register 1
ERXMISC2_EL1	3	C5	0	C5	2	0x0	64-bit	Selected Error Record Miscellaneous Register 2
ERXMISC3_EL1	3	C5	0	C5	3	0x0	64-bit	Selected Error Record Miscellaneous Register 3

12 GIC CPU interface

The *Generic Interrupt Controller* (GIC) supports and controls interrupts. The GIC distributor connects to the Neoverse™ N2 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 cluster has a GIC CPU interface which connects to a common external distributor component.

The GICv4.1 architecture implemented in the Neoverse™ N2 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.

12.1 Disable the GIC CPU interface

The Neoverse™ N2 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 **nFIQ** and **nIRQ** interrupt signals. If the Neoverse™ N2 core is not integrated with an external GIC interrupt distributor component in the system (minimum GIC v3 architecture), then you need to disable the GIC CPU interface.

If you disable the GIC CPU interface, then:

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

Note

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

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

12.2 AArch64 GIC register summary

The summary table provides an overview of all implementation defined GIC registers in the core. Individual register descriptions provide detailed information.

Table 12-1: GIC register summary

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

13 Advanced SIMD and floating-point support

The Neoverse™ N2 core supports the Advanced SIMD and scalar floating-point instructions in the A32, T32, A64 instruction sets without floating-point exception trapping. The Neoverse™ N2 core floating-point implementation includes Arm®v8.3-A and Arm®v8.5-A features.

The Neoverse™ N2 core implements all scalar operations in hardware with support for all combinations of:

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

14 Scalable Vector Extensions support

The Neoverse™ N2 core supports the *Scalable Vector Extension* (SVE) and the *Scalable Vector Extension 2* (SVE2). SVE and SVE2 complement and do not replace AArch64 Advanced SIMD and floating-point functionality.

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

The Neoverse™ N2 core implements a scalable vector length of 128 bits.

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

See the *Arm®v9-A Supplement for v8-A Arm® Architecture Reference Manual* for more information about SVE2.

15 System control

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

The main functions of the system registers are:

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

The system registers are accessible in both AArch32 execution state at EL0 only and AArch64 execution state at EL0 to EL3.

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

15.1 AArch64 identification register summary

The summary table provides an overview of all implementation defined identification registers in the core. Individual register descriptions provide detailed information.

Table 15-1: identification register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
MIDR_EL1	3	C0	0	C0	0	See individual bit resets.	64-bit	Main ID Register
MPIDR_EL1	3	C0	0	C0	5	See individual bit resets.	64-bit	Multiprocessor Affinity Register
REVIDR_EL1	3	C0	0	C0	6	See individual bit resets.	64-bit	Revision ID Register
ID_PFR0_EL1	3	C0	0	C1	0	See individual bit resets.	64-bit	AArch32 Processor Feature Register 0
ID_PFR1_EL1	3	C0	0	C1	1	See individual bit resets.	64-bit	AArch32 Processor Feature Register 1
ID_DFR0_EL1	3	C0	0	C1	2	See individual bit resets.	64-bit	AArch32 Debug Feature Register 0
ID_AFR0_EL1	3	C0	0	C1	3	`0x0`	64-bit	AArch32 Auxiliary Feature Register 0
ID_MMFR0_EL1	3	C0	0	C1	4	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 0
ID_MMFR1_EL1	3	C0	0	C1	5	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 1
ID_MMFR2_EL1	3	C0	0	C1	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 2
ID_MMFR3_EL1	3	C0	0	C1	7	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 3
ID_ISAR0_EL1	3	C0	0	C2	0	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 0
ID_ISAR1_EL1	3	C0	0	C2	1	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 1
ID_ISAR2_EL1	3	C0	0	C2	2	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 2
ID_ISAR3_EL1	3	C0	0	C2	3	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 3
ID_ISAR4_EL1	3	C0	0	C2	4	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 4

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
ID_ISAR5_EL1	3	C0	0	C2	5	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 5
ID_MMFR4_EL1	3	C0	0	C2	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 4
ID_ISAR6_EL1	3	C0	0	C2	7	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 6
MVFR0_EL1	3	C0	0	C3	0	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 0
MVFR1_EL1	3	C0	0	C3	1	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 1
MVFR2_EL1	3	C0	0	C3	2	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 2
ID_PFR2_EL1	3	C0	0	C3	4	See individual bit resets.	64-bit	AArch32 Processor Feature Register 2
ID_DFR1_EL1	3	C0	0	C3	5	0x0	64-bit	Debug Feature Register 1
ID_AA64PFR0_EL1	3	C0	0	C4	0	See individual bit resets.	64-bit	AArch64 Processor Feature Register 0
ID_AA64PFR1_EL1	3	C0	0	C4	1	See individual bit resets.	64-bit	AArch64 Processor Feature Register 1
ID_AA64ZFR0_EL1	3	C0	0	C4	4	See individual bit resets.	64-bit	SVE Feature ID register 0
ID_AA64DFR0_EL1	3	C0	0	C5	0	See individual bit resets.	64-bit	AArch64 Debug Feature Register 0
ID_AA64DFR1_EL1	3	C0	0	C5	1	0x0	64-bit	AArch64 Debug Feature Register 1
ID_AA64AFR0_EL1	3	C0	0	C5	4	0x0	64-bit	AArch64 Auxiliary Feature Register 0
ID_AA64AFR1_EL1	3	C0	0	C5	5	0x0	64-bit	AArch64 Auxiliary Feature Register 1
ID_AA64ISAR0_EL1	3	C0	0	C6	0	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 0
ID_AA64ISAR1_EL1	3	C0	0	C6	1	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 1
ID_AA64MMFR0_EL1	3	C0	0	C7	0	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 0
ID_AA64MMFR1_EL1	3	C0	0	C7	1	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 1
ID_AA64MMFR2_EL1	3	C0	0	C7	2	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 2
CLIDR_EL1	3	C0	1	C0	1	See individual bit resets.	64-bit	Cache Level ID Register
GMID_EL1	3	C0	1	C0	4	See individual bit resets.	64-bit	Multiple tag transfer ID register
CTR_EL0	3	C0	3	C0	1	See individual bit resets.	64-bit	Cache Type Register
DCZID_EL0	3	C0	3	C0	7	See individual bit resets.	64-bit	Data Cache Zero ID register
MPAMIDR_EL1	3	C10	0	C4	4	See individual bit resets.	64-bit	MPAM ID Register (EL1)
IMP_CPUCFR_EL1	3	C15	0	C0	0	See individual bit resets.	64-bit	CPU Configuration Register

16 Random number generator support

The Neoverse™ N2 core can be configured to support two random number instructions introduced in the Arm®v8.5-A extension.

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

- MRS Xn, `RNDR`
- MRS Xn, `RNDRRS`

The Neoverse™ N2 core expects the *True Random Number Generator* (TRNG) and the *Deterministic Random Bit Generator* (DRBG) to be available as a memory-mapped peripheral and must be capable of the following requirements.

- Design the TRNG and DRBG as architecturally stipulated.
- Provide as many copies of TRNG and DRBG as is necessary to meet the overall bandwidth and latency requirements of the system.
- Reseed the DRBG from the TRNG when a `RNDRRS` instruction is received, as defined by the address encoding described in the Neoverse™ N2 core microarchitecture.
- Provide *Quality of Service* (QOS) managed access to DRBG bandwidth as architecturally defined.
- Provide access to each TRNG and DRBG block through a memory-mapped Dev-nGnRnE read (LDP Xreg).
- The address used by the LDP Xreg for `RNDR` and `RNDRRS` instructions is a physical-address defined as follows:
    - A combination of base register of 64K page (CPURNDBR_EL3[47:16]), PE-specific identifier (CPURNDPEID_EL3[10:0]), instruction-type.
    - RNDR address: {CPURNDBR_EL3[47:16], CPURNDPEID_EL3[10:0], 1'b0, 4'b0}
    - RNDRRS address: {CPURNDBR_EL3[47:16], CPURNDPEID_EL3[10:0], 1'b1, 4'b0}
- Set CPURNDBR_EL3[47:16] in each Neoverse™ N2 core to match the peripheral base of the TRNG and DRBG block corresponding to the core. The association of the core to TRNG and DRBG block is defined by the system integrator.
- The TRNG and DRBG block must correctly decode the read-address, using [15:5] as the unique core identifier for QOS guarantees.
- The TRNG and DRBG block must correctly decode bit [4] of the read-address, 0 specifying an `RNDR` instruction and 1 specifying an `RNDRRS` instruction.
- Upon receiving a `RNDR` or `RNDRRS` request, the TRNG and DRBG block must return a 64-bit random number in the first 64 bits and 1 in the second 64 bits. In the event that the DRBG block is unable to provide a random number within a system integrator defined timeframe, it will return 0 in the first and second 64 bits.
- In the event of a bus error, a `RNDR` or `RNDRRS` request will fail and the core will set the PSTATE.Z flag and assert a SEI.

Note

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

16.1 AArch64 random number control register summary

The summary table provides an overview of all implementation defined random number control registers in the core. Individual register descriptions provide detailed information.

Table 16-1: random number control register summary

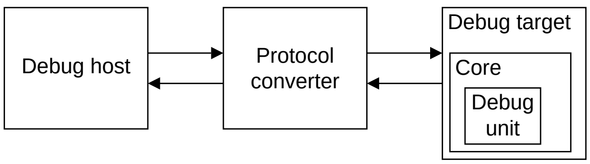
Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
IMP_CPURNDBR_EL3	3	C15	6	C3	0	0x0	64-bit	CPU Random Number Base Register
IMP_CPURNDPEID_EL3	3	C15	6	C3	1	0x0	64-bit	CPU Random Number Packet Identification Register

17 Debug

The Neoverse™ N2 core is part of a debug system and supports both self-hosted and external debug.

The following figure shows a typical external debug system:

Figure 17-1: External debug system



Each debug element is defined as follows:

Debug host

The debug host is a computer, for example, a personal computer that is running a software debugger, such as the Arm® Debugger . With the debug host, you can issue high-level commands, such as setting a breakpoint at a certain location or examining 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 using the *Advanced Peripheral Bus* (APB) slave interface. The debug target is, for example, a development system with a test chip or a silicon part with a core.

Debug unit

The debug unit helps debugging software that is running on the core, including:

- Operating systems
- Application software
- Hardware systems that are based on the core

With the debug unit, you can:

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

For self-hosted debug, the debug target runs additional debug monitor software that runs on the Neoverse™ N2 core itself. This way, it does not require expensive interface hardware to connect a second host computer.

17.1 Debug register interfaces

The Neoverse™ N2 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

5.6 Debug over powerdown on page 47

17.1.1 Core interfaces

System register access allows the Neoverse™ N2 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 *Advanced Peripheral Bus* (APB) slave port that connects into the DebugBlock of the DSU-110.

Performance monitoring

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

Trace

This function is system register based and memory-mapped. You can access the *Embedded Trace Macrocell* (ETM) 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 APB slave port interface, see *Interfaces* in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*.

17.1.2 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 *Embedded Trace Macrocell* (ETM) 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.

17.1.3 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.
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.

17.1.4 Breakpoints and watchpoints

The Neoverse™ N2 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.

17.2 Debug events

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

The Neoverse™ N2 core responds to a debug event in one of the following ways:

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

In the Neoverse™ N2 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.

17.3 Debug memory map and debug signals

The debug memory map and debug signals are handled at the DynamIQ™-110 cluster level.

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

17.4 ROM table

The Neoverse™ N2 core includes a ROM table that contains a list of components in the system. Debuggers can use the ROM table to determine which CoreSight components are implemented.

The ROM table is a CoreSight debug related component that aids system debug along with CoreSight SoC and is for the Neoverse™ N2 core. There is one ROM table for each core and ROM tables comply with the *Arm® CoreSight™ Architecture Specification v3.0*.

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

The Neoverse™ N2 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 ETM
0x000C	ROMENTRY3	Optional ELA

17.5 CoreSight component identification

Each component associated with the Neoverse™ N2 core has a unique set of CoreSight ID values.

Table 17-3: Neoverse™ N2 CoreSight component identification

Component	Peripheral ID	Component ID	DevType	DevArch	Revision
ETM	0x04000BBD49	0xB105900D	0x00000013	0x47705a13	r0p0
PMU	0x04000BBD49	0xB105900D	0x00000016	0x47702a16	r0p0
DBG	0x04000BBD49	0xB105900D	0x00000015	0x47709a15	r0p0
ROM Table	0x04000BBD49	0xB105900D	0x00000000	0x47700af7	r0p0

For details on the CoreSight component identification for the Neoverse™ N2 core ELA, see the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual*.

17.6 AArch64 debug register summary

The summary table provides an overview of all implementation defined debug registers in the core. Individual register descriptions provide detailed information.

Table 17-4: debug register summary

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

17.7 External Debug register summary

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

Table 17-5: External Debug register summary

Name	Reset	Width	Description
EDRCR	See individual bit resets.	32-bit	External Debug Reserve Control Register
EDACR	0x0	32-bit	External Debug Auxiliary Control Register
EDPRCR	See individual bit resets.	32-bit	External Debug Power/Reset Control Register
MIDR_EL1	See individual bit resets.	32-bit	Main ID Register
EDPFR	See individual bit resets.	64-bit	External Debug Processor Feature Register
EDDFR	See individual bit resets.	64-bit	External Debug Feature Register
EDDEVARCH	See individual bit resets.	32-bit	External Debug Device Architecture register
EDDEVID2	0x0	32-bit	External Debug Device ID register 2
EDDEVID1	See individual bit resets.	32-bit	External Debug Device ID register 1
EDDEVID	See individual bit resets.	32-bit	External Debug Device ID register 0
EDDEVTYPE	See individual bit resets.	32-bit	External Debug Device Type register
EDPIDR4	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 4
EDPIDR0	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 0
EDPIDR1	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 1
EDPIDR2	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 2
EDPIDR3	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 3
EDCIDR0	See individual bit resets.	32-bit	External Debug Component Identification Register 0
EDCIDR1	See individual bit resets.	32-bit	External Debug Component Identification Register 1
EDCIDR2	See individual bit resets.	32-bit	External Debug Component Identification Register 2
EDCIDR3	See individual bit resets.	32-bit	External Debug Component Identification Register 3

17.8 External CoreROM register summary

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

Table 17-6: External CoreROM register summary

Name	Reset	Width	Description
COREROM_ROMENTRY0	See individual bit resets.	32-bit	Core ROM table Entry 0
COREROM_ROMENTRY1	See individual bit resets.	32-bit	Core ROM table Entry 1
COREROM_ROMENTRY2	See individual bit resets.	32-bit	Core ROM table Entry 2
COREROM_ROMENTRY3	See individual bit resets.	32-bit	Core ROM table Entry 3
COREROM_AUTHSTATUS	See individual bit resets.	32-bit	Core ROM table Authentication Status Register
COREROM_DEVARCH	See individual bit resets.	32-bit	Core ROM table Device Architecture Register

Name	Reset	Width	Description
COREROM_DEVTYPE	See individual bit resets.	32-bit	Core ROM table Device Type Register
COREROM_PIDR4	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 4
COREROM_PIDR0	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 0
COREROM_PIDR1	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 1
COREROM_PIDR2	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 2
COREROM_PIDR3	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 3
COREROM_CIDR0	See individual bit resets.	32-bit	Core ROM table Component Identification Register 0
COREROM_CIDR1	See individual bit resets.	32-bit	Core ROM table Component Identification Register 1
COREROM_CIDR2	See individual bit resets.	32-bit	Core ROM table Component Identification Register 2
COREROM_CIDR3	See individual bit resets.	32-bit	Core ROM table Component Identification Register 3

18 Performance Monitors Extension support

The Neoverse™ N2 core implements the Performance Monitors Extension, including Arm®v8.4-A and Arm®v8.5-A performance monitoring features.

The Neoverse™ N2 core *Performance Monitoring Unit* (PMU):

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

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

18.1 Performance monitors events

The Neoverse™ N2 *Performance Monitoring Unit* (PMU) collects events from other units in the design and uses numbers to reference these events.

The following table lists the Neoverse™ N2 performance monitors events. Event reference numbers that are not listed are reserved.

Note

Unless otherwise indicated, each of these events can be exported to the *Embedded Trace Macrocell* (ETM).

Table 18-1: Performance monitors Events

Event number	Event mnemonic	Event description
`0x0`	SW_INCR	Software increment This event counts any instruction architecturally executed (condition code check pass).

Event number	Event mnemonic	Event description
`0x1`	L1I_CACHE_REFILL	L1 instruction cache refill This event counts any instruction fetch which misses in the cache. The following instructions are not counted: • Cache maintenance instructions • Non-cacheable accesses
`0x2`	L1I_TLB_REFILL	L1 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.
`0x3`	L1D_CACHE_REFILL	L1 data cache refill This event counts any load or store operation or translation table walk access which causes data to be read from outside the L1, including accesses which do not allocate into L1. 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.
`0x4`	L1D_CACHE	L1 data cache access This event counts any load or store operation or translation table walk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL event causes this event to count. 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.
`0x5`	L1D_TLB_REFILL	L1 data TLB refill This event counts any refill of the data L1 TLB from the MMUTC. This includes refills that result in a translation fault. TLB maintenance instructions are not counted. This event counts regardless of whether the MMU is enabled.
`0x8`	INST_RETIRED	Instruction architecturally executed. This event counts all retired instructions, including those that fail their condition check.

Event number	Event mnemonic	Event description
0x9	EXC_TAKEN	Exception taken
0x0A	EXC_RETURN	Instruction architecturally executed, condition code check pass, exception return
0x0B	CID_WRITE_RETIRED	Instruction architecturally executed, condition code check pass, write to CONTEXTIDR This event only counts writes to CONTEXTIDR_EL1. Writes to CONTEXTIDR_EL12 and CONTEXTIDR_EL2 are not counted.
0x10	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed This event counts any predictable branch instruction which is mispredicted either because of dynamic misprediction or because the MMU is off and the branches are statically predicted not taken.
0x11	CPU_CYCLES	Cycle This event is not exported to the ETM.
0x12	BR_PRED	Predictable branch speculatively executed. This event counts all predictable branches.
0x13	MEM_ACCESS	Data memory access. This event counts memory accesses due to load or store instructions. The following instructions are not counted: • Instruction fetches • Cache maintenance instructions • Translation table walks or prefetches This event counts the sum of MEM_ACCESS_RD and MEM_ACCESS_WR.
0x14	L1I_CACHE	L1 instruction cache access or *Macro-op* (MOP) cache access. This event counts any instruction fetch which accesses the L1 instruction cache or MOP cache. The following instructions are not counted: • Cache maintenance instructions • Non-cacheable accesses
0x15	L1D_CACHE_WB	L1 data cache Write-Back This event counts any write-back of data from the L1 data cache to L2 or L3. This counts both victim line evictions and snoops, including cache maintenance operations. The following instructions are not counted: • 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

Event number	Event mnemonic	Event description
`0x16`	L2D_CACHE	L2 cache access This event counts any transaction from L1 which looks up in the L2 cache, and any write-back from the L1 to the L2. Snoops from outside the core and cache maintenance operations are not counted.
`0x17`	L2D_CACHE_REFILL	L2 cache refill This event counts any cacheable transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 are not counted.
`0x18`	L2D_CACHE_WB	L2 cache write-back This event counts any write-back of data from the L2 cache to outside the core. This includes snoops to the L2 which return data, regardless of whether they cause an invalidation. Invalidations from the L2 which do not write data outside of the core and snoops which return data from the L1 are not counted.
`0x19`	BUS_ACCESS	Bus access This event counts for every beat of data transferred over the data channels between the core and the *Snoop Control Unit* (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, BUS_ACCESS_WR, and any snoop data responses.
`0x1A`	MEMORY_ERROR	Local memory error This event counts any correctable or uncorrectable memory error (ECC or parity) in the protected core RAMs.
`0x1B`	INST_SPEC	Operation speculatively executed
`0x1C`	TTBR_WRITE_RETIRED	Instruction architecturally executed, condition code check pass, write to TTBR This event only counts writes to TTBR0_EL1/TTBR1_EL1. Accesses to TTBR0_EL12/TTBR1_EL12 or TTBR0_EL2/TTBR1_EL2 are not counted.
`0x1D`	BUS_MASTER_CYCLE	Bus cycles This event duplicates CPU_CYCLES.
`0x1E`	COUNTER_OVERFLOW	For odd-numbered counters, this event increments the count by one for each overflow of the preceding even-numbered counter. For even-numbered counters, there is no increment.
`0x20`	CACHE_ALLOCATE	L2 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.

Event number	Event mnemonic	Event description
0x21	BR_RETIRED	Instruction architecturally executed, branch This event counts all branches, taken or not. This excludes exception entries, debug entries and CCFAIL branches.
0x22	BR_MIS_PRED_RETIRED	Instruction architecturally executed, mispredicted branch This event counts any branch counted by BR_RETIRED which is not correctly predicted and causes a pipeline flush.
0x23	STALL_FRONTEND	No operation issued because of the frontend The counter counts on any cycle when there are no fetched instructions available to dispatch.
0x24	STALL_BACKEND	No operation issued because of the backend The counter counts on any cycle fetched instructions are not dispatched due to resource constraints.
0x25	L1D_TLB	Level 1 data TLB access This event counts any load or store operation which accesses the data L1 TLB. If both a load and a store are executed on a cycle, this event counts twice. This event counts regardless of whether the MMU is enabled.
0x26	L1I_TLB	Level 1 instruction TLB access This event counts any instruction fetch which accesses the instruction L1 TLB. This event counts regardless of whether the MMU is enabled.
0x29	L3D_CACHE_ALLOCATE	Attributable L3 cache allocation without refill This event counts any full cache line write into the L3 cache which does not cause a linefill, including write-backs from L2 to L3 and full-line writes which do not allocate into L2.
0x2A	L3D_CACHE_REFILL	Attributable L3 cache refill This event counts for any cacheable read transaction returning data from the SCU for which the data source was outside the cluster. Transactions such as ReadUnique are counted as read transactions, even though they can be generated by store instructions.
0x2B	L3D_CACHE	Attributable L3 cache access This event counts for any cacheable read transaction returning data from the SCU, or for any cacheable write to the SCU.
0x2D	L2TLB_REFILL	Attributable L2 TLB refill This event counts on any refill of the MMUTC, caused by either an instruction or data access. This event does not count if the MMU is disabled.

Event number	Event mnemonic	Event description
0x2F	L2TLB_REQ	Attributable L2 TLB access This event counts on any access to the MMUTC (caused by a refill of any of the L1 TLBs). This event does not count if the MMU is disabled.
0x31	REMOTE_ACCESS	Access to another socket in a multi-socket system
0x34	DTLB_WLK	Access to data TLB that caused a page table walk This event counts on any data access which causes L2D_TLB_REFILL to count.
0x35	ITLB_WLK	Access to instruction TLB that caused a translation table walk. This event counts on any instruction access which causes L2D_TLB_REFILL to count.
0x36	LL_CACHE_RD	Last level cache access, read If CPUECTLR.EXTLLC is set, then this event counts any cacheable read transaction which returns a data source of interconnect cache. If CPUECTLR.EXTLLC is not set, then this event is a duplicate of the L*D_CACHE_RD event corresponding to the last level of cache implemented L2D_CACHE_RD if only one is implemented, or L1D_CACHE_RD if neither is implemented.
0x37	LL_CACHE_MISS_RD	Last level cache miss, read If CPUECTLR.EXTLLC is set, then this event counts any cacheable read transaction which returns a data source of DRAM, remote, or inter-cluster peer. If CPUECTLR.EXTLLC is not set, then this event is a duplicate of the L*D_CACHE_REFILL_RD event corresponding to the last level of cache implemented L2D_CACHE_REFILL_RD if only one is implemented, or L1D_CACHE_REFILL_RD if neither is implemented.
0x39	L1D_CACHE_LMISS_RD	Level 1 data cache long-latency miss
0x3A	OP_RETIRED	Micro-operation architecturally executed
0x3B	OP_SPEC	Micro-operation speculatively executed
0x3C	STALL	No operation sent for execution
0x3D	STALL_SLOT_BACKEND	No operation sent for execution on a slot due to the backend
0x3E	STALL_SLOT_FRONTEND	No operation sent for execution on a slot due to the frontend
0x3F	STALL_SLOT	No operation sent for execution on a slot
0x40	L1D_CACHE_RD	L1 data cache access, read This event counts any load operation or page table walk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_RD event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses

Event number	Event mnemonic	Event description
`0x41`	L1D_CACHE_WR	L1 data cache access, write This event counts any store operation which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_WR event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses
`0x42`	L1D_CACHE_REFILL_RD	L1 data cache refill, read This event counts any load operation or page table walk access which causes data to be read from outside the L1, including accesses which do not allocate into L1. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses
`0x43`	L1D_CACHE_REFILL_WR	L1 data cache refill, write This event counts any store operation which causes data to be read from outside the L1, 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
`0x44`	L1D_CACHE_REFILL_INNER	L1 data cache refill, inner This event counts any L1 data cache linefill (as counted by L1D_CACHE_REFILL) which hits in the L2 cache, system L3 cache, or another core in the cluster.
`0x45`	L1D_CACHE_REFILL_OUTER	L1 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, system L3 cache, or another core in the cluster, and instead obtains data from outside the cluster.
`0x46`	L1D_CACHE_WB_VICTIM	L1 data cache write-back, victim
`0x47`	L1D_CACHE_WB_CLEAN	L1 data cache write-back cleaning and coherency
`0x48`	L1D_CACHE_INVAL	L1 data cache invalidate
`0x4C`	L1D_TLB_REFILL_RD	L1 data TLB refill, read
`0x4D`	L1D_TLB_REFILL_WR	L1 data TLB refill, write
`0x4E`	L1D_TLB_RD	L1 data TLB access, read
`0x4F`	L1D_TLB_WR	L1 data TLB access, write

Event number	Event mnemonic	Event description
0x50	CACHE_ACCESS_RD	L2 cache access, read This event counts any read transaction from L1 which looks up in the L2 cache. Snoops from outside the core are not counted.
0x51	CACHE_ACCESS_WR	L2 cache access, write This event counts any write transaction from L1 which looks up in the L2 cache or any write-back from L1 which allocates into the L2 cache. Snoops from outside the core are not counted.
0x52	CACHE_RD_REFILL	L2 cache refill, read This event counts any cacheable read transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 should not be counted. Transactions such as ReadUnique are counted as read transactions, even though they can be generated by store instructions.
0x53	CACHE_WR_REFILL	L2 cache refill, write This event counts any write transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 should not be counted. Transactions such as ReadUnique are not counted as write transactions.
0x56	CACHE_WRITEBACK_VICTIM	L2 cache write-back, victim
0x57	CACHE_WRITEBACK_CLEAN_COH	L2 cache write-back, cleaning and coherency
0x58	L2CACHE_INV	L2 cache invalidate
0x5C	L2TLB_RD_REFILL	L2 TLB refill, read
0x5D	L2TLB_WR_REFILL	L2 TLB refill, write
0x5E	L2TLB_RD_REQ	L2 TLB access, read
0x5F	L2TLB_WR_REQ	L2 TLB access, write
0x60	BUS_ACCESS_RD	Bus access read This event counts for every beat of data transferred over the read data channel between the core and the SCU.
0x61	BUS_ACCESS_WR	Bus access write This event counts for every beat of data transferred over the write data channel between the core and the SCU.

Event number	Event mnemonic	Event description
0x66	MEM_ACCESS_RD	Data memory access, read This event counts memory accesses due to load instructions. The following instructions are not counted: • Instruction fetches • Cache maintenance instructions • Translation table walks • Prefetches
0x67	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
0x68	UNALIGNED_LD_SPEC	Unaligned access, read
0x69	UNALIGNED_ST_SPEC	Unaligned access, write
0x6A	UNALIGNED_LDST_SPEC	Unaligned access
0x6C	LDREX_SPEC	Exclusive operation speculatively executed, LDREX or LDX
0x6D	STREX_PASS_SPEC	Exclusive operation speculatively executed, STREX or STX pass
0x6E	STREX_FAIL_SPEC	Exclusive operation speculatively executed, STREX or STX fail
0x6F	STREX_SPEC	Exclusive operation speculatively executed, STREX or STX
0x70	LD_SPEC	Operation speculatively executed, load
0x71	ST_SPEC	Operation speculatively executed, store
0x73	DP_SPEC	Operation speculatively executed, integer data-processing
0x74	ASE_SPEC	Operation speculatively executed, Advanced SIMD instruction
0x75	VFP_SPEC	Operation speculatively executed, floating-point instruction
0x76	PC_WRITE_SPEC	Operation speculatively executed, software change of the PC
0x77	CRYPTO_SPEC	Operation speculatively executed, Cryptographic instruction
0x78	BR_IMMED_SPEC	Branch speculatively executed, immediate branch
0x79	BR_RETURN_SPEC	Branch speculatively executed, procedure return
0x7A	BR_INDIRECT_SPEC	Branch speculatively executed, indirect branch
0x7C	ISB_SPEC	Barrier speculatively executed, ISB
0x7D	DSB_SPEC	Barrier speculatively executed, DSB
0x7E	DMB_SPEC	Barrier speculatively executed, DMB
0x81	EXC_UNDEF	Counts the number of undefined exceptions taken locally

Event number	Event mnemonic	Event description
0x82	EXC_SVC	Exception taken locally, Supervisor Call
0x83	EXC_PABORT	Exception taken locally, Instruction Abort
0x84	EXC_DABORT	Exception taken locally, Data Abort and SError
0x86	EXC_IRQ	Exception taken locally, IRQ
0x87	EXC_FIQ	Exception taken locally, FIQ
0x88	EXC_SMC	Exception taken locally, Secure Monitor Call
0x8A	EXC_HVC	Exception taken locally, Hypervisor Call
0x8B	EXC_TRAP_PABORT	Exception taken, Instruction Abort not taken locally
0x8C	EXC_TRAP_DABORT	Exception taken, Data Abort or SError not taken locally
0x8D	EXC_TRAP_OTHER	Exception taken, Other traps not taken locally
0x8E	EXC_TRAP_IRQ	Exception taken, IRQ not taken locally
0x8F	EXC_TRAP_FIQ	Exception taken, FIQ not taken locally
0x90	RC_LD_SPEC	Release consistency operation speculatively executed, load-acquire
0x91	RC_ST_SPEC	Release consistency operation speculatively executed, store-release
0xA0	L3_CACHE_RD	L3 cache read
0x4000	SAMPLE_POP	Sample population
0x4001	SAMPLE_FEED	Sample taken
0x4002	SAMPLE_FILTRATE	Sample taken and not removed by filtering
0x4003	SAMPLE_COLLISION	Sample collided with previous sample
0x4004	CNT_CYCLES	Constant frequency cycles
0x4005	STALL_BACKEND_MEM	No operation sent due to the backend and memory stalls
0x4006	L1I_CACHE_LMISS	L1 instruction cache long latency miss
0x4009	L2D_CACHE_LMISS_RD	L2 cache long latency miss
0x400B	L3D_CACHE_LMISS_RD	L3 cache long latency miss
0x400C	TRB_WRAP	Trace buffer current write pointer wrapped
0x4010	TRCEXTOUT0	PE Trace Unit external output 0
0x4011	TRCEXTOUT1	PE Trace Unit external output 1
0x4012	TRCEXTOUT2	PE Trace Unit external output 2
0x4013	TRCEXTOUT3	PE Trace Unit external output 3
0x4018	CTI_TRIGOUT4	Cross-trigger Interface output trigger 4
0x4019	CTI_TRIGOUT5	Cross-trigger Interface output trigger 5
0x401A	CTI_TRIGOUT6	Cross-trigger Interface output trigger 6
0x401B	CTI_TRIGOUT7	Cross-trigger Interface output trigger 7
0x4020	LDST_ALIGN_LAT	Access with additional latency from alignment
0x4021	LD_ALIGN_LAT	Load with additional latency from alignment
0x4022	ST_ALIGN_LAT	Store with additional latency from alignment
0x4024	MEM_ACCESS_CHECKED	Checked data memory access
0x4025	MEM_ACCESS_RD_CHECKED	Checked data memory access, read
0x4026	MEM_ACCESS_WR_CHECKED	Checked data memory access, write
0x8005	ASE_INST_SPEC	Advanced SIMD operations speculatively executed

Event number	Event mnemonic	Event description
0x8006	SVE_INST_SPEC	SVE operations speculatively executed
0x8014	FP_HP_SPEC	Half-precision floating-point operation speculatively executed
0x8018	FP_SP_SPEC	Single-precision floating-point operation speculatively executed
0x801C	FP_DP_SPEC	Double-precision floating-point operation speculatively executed
0x8074	SVE_PRED_SPEC	SVE predicated operations speculatively executed
0x8075	SVE_PRED_EMPTY_SPEC	SVE predicated operations with no active predicates speculatively executed
0x8076	SVE_PRED_FULL_SPEC	SVE predicated operations speculatively executed with all active predicates
0x8077	SVE_PRED_PARTIAL_SPEC	SVE predicated operations speculatively executed with partially active predicates
0x8079	SVE_PRED_NOT_FULL_SPEC	SVE predicated operations speculatively executed with a Governing predicate in which at least one element is FALSE
0x80BC	SVE_LDFF_SPEC	SVE First-fault load operations speculatively executed
0x80BD	SVE_LDFF_FAULT_SPEC	SVE First-fault load operations speculatively executed which set FFR bit to 0
0x80C0	FP_SCALE_OPS_SPEC	Scalable floating-point element operations speculatively executed
0x80C1	FP_FIXED_OPS_SPEC	Non-scalable floating-point element operations speculatively executed
0x80E3	ASE_SVE_INT8_SPEC	Operation counted by ASE_SVE_INT_SPEC where the largest type is 8-bit integer
0x80E7	ASE_SVE_INT16_SPEC	Operation counted by ASE_SVE_INT_SPEC where the largest type is 16-bit integer
0x80EB	ASE_SVE_INT32_SPEC	Operation counted by ASE_SVE_INT_SPEC where the largest type is 32-bit integer
0x80EF	ASE_SVE_INT64_SPEC	Operation counted by ASE_SVE_INT_SPEC where the largest type is 64-bit integer

18.2 Performance monitors interrupts

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 Neoverse™ N2 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 Armv8, for Armv8-A architecture profile. The register descriptions provided in this manual describe whether each register is read/write or read-only.

18.4 AArch64 performance monitors register summary

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

Table 18-2: performance-monitors register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
PMMIR_EL1	3	C9	0	C14	6	See individual bit resets.	64-bit	Performance Monitors Machine Identification Register
PMCR_EL0	3	C9	3	C12	0	See individual bit resets.	64-bit	Performance Monitors Control Register
PMCEID0_EL0	3	C9	3	C12	6	See individual bit resets.	64-bit	Performance Monitors Common Event Identification register 0
PMCEID1_EL0	3	C9	3	C12	7	See individual bit resets.	64-bit	Performance Monitors Common Event Identification register 1

18.5 External Performance monitors register summary

The summary table provides an overview of all External Performance monitors registers in the core. Individual register descriptions provide detailed information.

Table 18-3: External Performance monitors register summary

Offset	Name	Reset	Width	Description
0x600	PMPCSSR	See individual bit resets.	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x60C	PMCID2SSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register
0x610	PMSSSR	`0x1`	32-bit	PMU Snapshot Status Register
0x618	PMCCNTSR	See individual bit resets.	64-bit	PMU Cycle Counter Snapshot Register
0x620	PMEVCNTSR0	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x628	PMEVCNTSR1	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x630	PMEVCNTSR2	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x638	PMEVCNTSR3	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x640	PMEVCNTSR4	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x648	PMEVCNTSR5	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F0	PMSSCR	See individual bit resets.	32-bit	PMU Snapshot Capture Register
0xE00	PMCFGR	See individual bit resets.	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_EL0	See individual bit resets.	32-bit	Performance Monitors Control Register
0xE20	PMCEID0	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 0

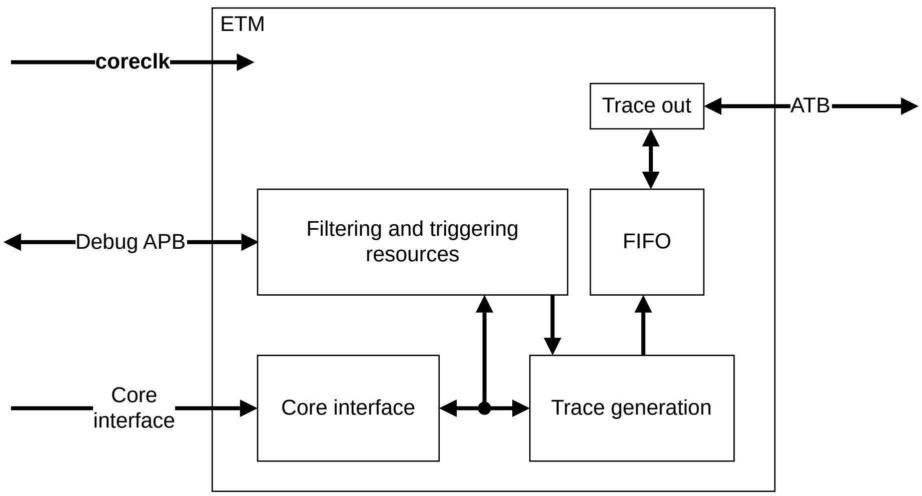
Offset	Name	Reset	Width	Description
0xE24	PMCEID1	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 1
0xE28	PMCEID2	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 2
0xE2C	PMCEID3	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 3
0xE40	PMMIR	See individual bit resets.	32-bit	Performance Monitors Machine Identification Register
0xFBC	PMDEVARCH	See individual bit resets.	32-bit	Performance Monitors Device Architecture register
0xFC8	PMDEVID	See individual bit resets.	32-bit	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	See individual bit resets.	32-bit	Performance Monitors Device Type register
0xFD0	PMPIDR4	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 3

19 Embedded Trace Extension support

The Neoverse™ N2 core implements the *Embedded Trace Extension* (ETE) with an *Embedded Trace Macrocell* (ETM). The ETM performs real-time instruction flow tracing based on the ETE. The ETM 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 ETM:

Figure 19-1: ETM 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 ETM generates by filtering. For example, you can limit trace generation to a certain address range. The ETM supports other, more complicated, logic analyzer style filtering options. The ETM can also generate a trigger that is a signal to the Trace Capture Device to stop capturing trace.

FIFO

The ETM 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® Embedded Trace Extension* for more information.

19.1 Trace unit resources

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

The following table shows the *Embedded Trace Macrocell* (ETM) trace unit resources, and indicates which of these resources the N2 core implements.

Table 19-1: ETM trace unit resources implemented

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

See the *Arm® Embedded Trace Extension* for more information.

19.2 Trace unit generation options

The Neoverse™ N2 core *Embedded Trace Macrocell* (ETM) trace unit implements a set of generation options.

The following table shows the trace generation options that are implemented in the Neoverse™ N2 core ETM trace unit.

Table 19-2: ETM trace unit generation options implemented

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

Description	Configuration
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® Embedded Trace Extension* for more information.

19.3 Reset the Embedded Trace Macrocell

The reset for the *Embedded Trace Macrocell* (ETM) is the same as a Cold reset for the core. In *TRace Buffer Extension* (TRBE) mode, a Warm reset keeps the TRBE disabled and therefore trace is not possible during Warm reset.

If the ETM trace unit is reset, then tracing stops until the ETM 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 Embedded Trace Macrocell registers

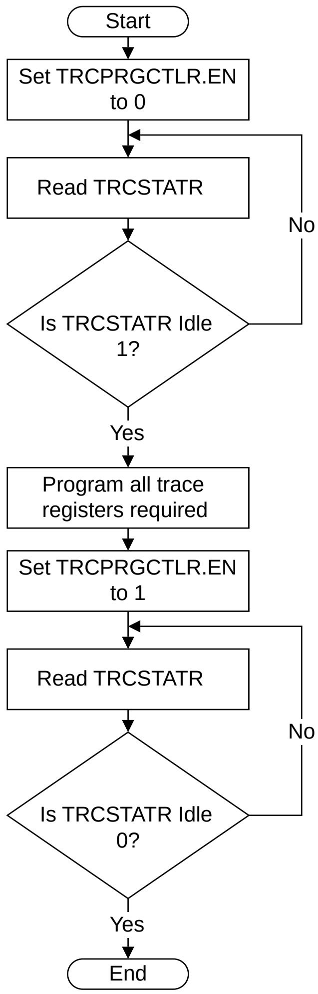
You program and read the *Embedded Trace Macrocell* (ETM) registers using either the Debug APB interface or the System register interface.

The core does not have to be in debug state when you program the ETM registers. When you program the ETM 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 ETM trace unit, use the TRCPRGCTLR.EN bit. See the *Arm® Embedded Trace Macrocell Architecture Specification* for more information about the following registers:

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

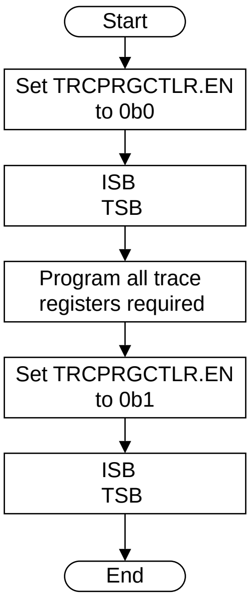
The following figure shows the flow for programming ETM registers using the Debug APB interface:

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



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

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



19.5 Embedded Trace Macrocell register interfaces

The Neoverse™ N2 core supports an APB memory-mapped interface and a system register interface to *Embedded Trace Macrocell* (ETM) registers.

Register accesses differ depending on the ETM state. See the *Arm® Embedded Trace Extension* for information on the behaviors and access mechanisms.

19.6 Interaction with the Performance Monitoring Unit and Debug

The *Embedded Trace Macrocell* (ETM) interacts with the *Performance Monitoring Unit* (PMU) and it can access the PMU events.

Interaction with the PMU

The Neoverse™ N2 core includes a PMU that enables events, such as cache misses and executed instructions, to be counted over time.

The PMU and ETM trace unit function together.

Use of PMU events by the ETM trace unit

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

The ETM 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 ETM trace unit.

Related information

18 Performance Monitors Extension support on page 105
18.1 Performance monitors events on page 105

19.7 ETE events

The Neoverse™ N2 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 105, the following events are also exported:

Table 19-3: ETE events

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

19.8 AArch64 trace register summary

The summary table provides an overview of all implementation defined trace registers in the core. Individual register descriptions provide detailed information.

Table 19-4: trace register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
TRCIDR8	2	C0	1	C0	6	See individual bit resets.	64-bit	ID Register 8
TRCIMSPEC0	2	C0	1	C0	7	See individual bit resets.	64-bit	IMP DEF Register 0
TRCIDR2	2	C0	1	C10	7	See individual bit resets.	64-bit	ID Register 2
TRCIDR3	2	C0	1	C11	7	See individual bit resets.	64-bit	ID Register 3
TRCIDR4	2	C0	1	C12	7	See individual bit resets.	64-bit	ID Register 4
TRCIDR5	2	C0	1	C13	7	See individual bit resets.	64-bit	ID Register 5
TRCIDR10	2	C0	1	C2	6	0x0	64-bit	ID Register 10
TRCIDR11	2	C0	1	C3	6	0x0	64-bit	ID Register 11

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
TRCIDR12	2	C0	1	C4	6	0x0	64-bit	ID Register 12
TRCIDR13	2	C0	1	C5	6	0x0	64-bit	ID Register 13
TRCIDR0	2	C0	1	C8	7	See individual bit resets.	64-bit	ID Register 0
TRCIDR1	2	C0	1	C9	7	See individual bit resets.	64-bit	ID Register 1
TRCCIDCVR0	2	C3	1	C0	0	See individual bit resets.	64-bit	Context Identifier Comparator Value Registers <n>

19.9 External Trace register summary

The summary table provides an overview of all external Trace registers in the core. Individual register descriptions provide detailed information.

Table 19-5: External Trace register summary

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

Offset	Name	Reset	Width	Description
0xFDC	TRCPIDR7	`0x0`	32-bit	Peripheral Identification Register 7
0xFE0	TRCPIDR0	See individual bit resets.	32-bit	Peripheral Identification Register 0
0xFE4	TRCPIDR1	See individual bit resets.	32-bit	Peripheral Identification Register 1
0xFE8	TRCPIDR2	See individual bit resets.	32-bit	Peripheral Identification Register 2
0xFEC	TRCPIDR3	See individual bit resets.	32-bit	Peripheral Identification Register 3
0xFF0	TRCCIDR0	See individual bit resets.	32-bit	Component Identification Register 0
0xFF4	TRCCIDR1	See individual bit resets.	32-bit	Component Identification Register 1
0xFF8	TRCCIDR2	See individual bit resets.	32-bit	Component Identification Register 2
0xFFC	TRCCIDR3	See individual bit resets.	32-bit	Component Identification Register 3

20 Trace Buffer Extension support

The Neoverse™ N2 core implements the *TRace Buffer Extension* (TRBE). The TRBE writes the program flow trace generated by the *Embedded Trace Macrocell* (ETM) trace unit directly to memory. The TRBE is programmed through System registers.

When enabled, the TRBE can:

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

When disabled, the TRBE ignores trace data and the ETM 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.

20.2 Trace buffer register interface

The Neoverse™ N2 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 Armv8, for Armv8-A architecture profile* for information on the behaviors and access mechanisms.

21 Activity Monitors Extension support

The Neoverse™ N2 core implements the Activity Monitors Extension to the Arm®v8.4-A architecture. Activity monitoring has features similar to performance monitoring features, but is intended for system management use whereas performance monitoring is aimed at user and debug applications.

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

The Neoverse™ N2 core implements seven counters in two groups, each of which is a 64-bit counter that counts a fixed event. Group 0 has four counters 0-3, and Group 1 has three counters 0-2.

21.1 Activity monitors access

The Neoverse™ N2 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 Armv8, for Armv8-A architecture profile* 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 Armv8, for Armv8-A architecture profile*.

System register access

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

External memory-mapped access

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

Base address for AMU registers on the utility bus interface is 0x<n>90000 where “n” is the Neoverse™ N2 core instance number in the DSU-110 cluster.

21.2 Activity monitors counters

The Neoverse™ N2 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 Neoverse™ N2 core are either fixed or programmable, and they map to the activity monitors counters.

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

Table 21-1: Mapping of counters to fixed events

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR00	CPU_CYCLES	`0x0011`	Core frequency cycles
AMEVCNTR01	CNT_CYCLES	`0x4004`	Constant frequency cycles
AMEVCNTR02	Instructions retired	`0x0008`	Instruction architecturally executed This counter increments for every instruction that is executed architecturally, including instructions that fail their condition code check.
AMEVCNTR03	STALL_BACKEND_MEM	`0x4005`	Memory stall cycles This counter counts cycles in which the core is unable to dispatch instructions from the front end to the back end due to a back end stall caused by a miss in the last level of cache within the core clock domain.
AMEVCNTR10	Reserved	`0x0300`	Reserved
AMEVCNTR11	Reserved	`0x0301`	Reserved
AMEVCNTR12	Reserved	`0x0302`	Reserved

21.4 AArch64 activity monitors register summary

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

Table 21-2: activity monitors register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
AMEVTYPER10_EL0	3	C13	3	C14	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	C13	3	C14	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	C13	3	C14	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMCFGR_EL0	3	C13	3	C2	1	See individual bit resets.	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	3	C13	3	C2	2	See individual bit resets.	64-bit	Activity Monitors Counter Group Configuration Register
AMEVTYPER00_EL0	3	C13	3	C6	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	C13	3	C6	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	C13	3	C6	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	C13	3	C6	3	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0

21.5 External Activity monitors register summary

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

Table 21-3: External Activity monitors register summary

Offset	Name	Reset	Width	Description
0x400	AMEVTYPER00	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x480	AMEVTYPER10	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x484	AMEVTYPER11	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x488	AMEVTYPER12	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x48C	AMEVTYPER13	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0xCE0	AMCGCR	See individual bit resets.	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	See individual bit resets.	32-bit	Activity Monitors Configuration Register
0xE08	AMIIDR	See individual bit resets.	32-bit	Activity Monitors Implementation Identification Register

Offset	Name	Reset	Width	Description
0xFBC	AMDEVARCH	See individual bit resets.	32-bit	Activity Monitors Device Architecture Register
0xFCC	AMDEVTYPE	See individual bit resets.	32-bit	Activity Monitors Device Type Register
0xFD0	AMPIDR4	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 4
0xFE0	AMPIDR0	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 0
0xFE4	AMPIDR1	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 1
0xFE8	AMPIDR2	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 2
0xFEC	AMPIDR3	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 3
0xFF0	AMCIDR0	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 0
0xFF4	AMCIDR1	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 1
0xFF8	AMCIDR2	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 2
0xFFC	AMCIDR3	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 3

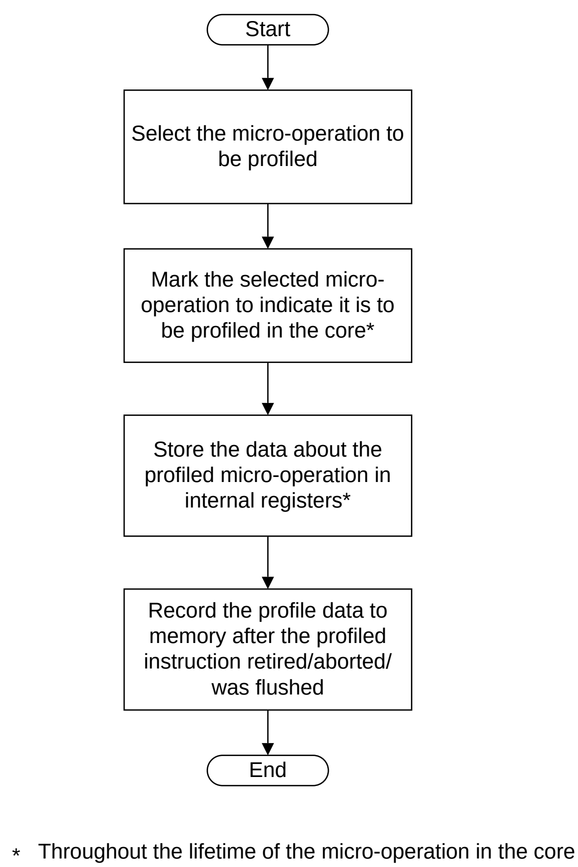
22 Statistical Profiling Extension support

The Neoverse™ N2 core implements the optional *Statistical Profiling Extension* (SPE) to the Arm®v8.5-A architecture. The SPE provides a statistical view of the performance characteristics of executed instructions that software writers can use to optimize their code for better performance.

The Neoverse™ N2 core profiles micro-operations to minimize the amount of logic necessary to support the SPE.

The following figure shows the SPE behavior in the Neoverse™ N2 core.

Figure 22-1: SPE behavior



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

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

Note

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

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

22.1 Statistical Profiling Extension events packet

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

The following table shows the events defined in the 32-bit events packet implemented in the Neoverse™ N2 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 Neoverse™ N2 core.

Table 22-2: SPE data source packet

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

22.3 Statistical profiling register interface

The Neoverse™ N2 core supports a System register interface to *Statistical Profiling Extension* (SPE) registers.

Register accesses differ depending on the SPE state. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on the behaviors and access mechanisms.

22.4 AArch64 statistical profiling-extension register summary

The summary table provides an overview of all implementation defined statistical profiling extension registers in the core. Individual register descriptions provide detailed information.

Table 22-3: statistical profiling extension register summary

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

Appendix A AArch32 registers

This appendix contains the descriptions for the Neoverse™ N2 AArch32 registers.

A.1 generic-system-control register summary

The summary table provides an overview of all implementation defined generic-system-control registers in the core. Individual register descriptions provide detailed information.

Table A-1: generic-system-control register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
FPSCR	3	C0	0	C4	1	See individual bit resets.	32-bit	Floating-Point Status and Control Register

A.1.1 FPSCR, Floating-Point Status and Control Register

Provides floating-point system status information and control.

Configurations

The named fields in this register map to the equivalent fields in the AArch64 AArch64-FPCR and AArch64-FPSR.

It is IMPLEMENTATION DEFINED whether the Len and Stride fields can be programmed to non-zero values, which will cause some AArch32 floating-point instruction encodings to be UNDEFINED, or whether these fields are RAZ.

Implemented only if the implementation includes the Advanced SIMD and floating-point functionality.

Attributes

Width

32

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure A-1: AArch32_fpscr bit assignments

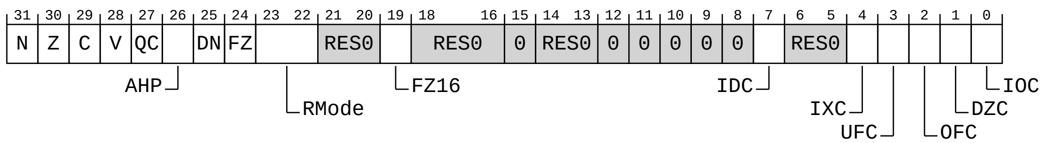


Table A-2: FPSCR bit descriptions

Bits	Name	Description	Reset
[31]	N	Negative condition flag. This is updated by floating-point comparison operations.	
[30]	Z	Zero condition flag. This is updated by floating-point comparison operations.	
[29]	C	Carry condition flag. This is updated by floating-point comparison operations.	
[28]	V	Overflow condition flag. This is updated by floating-point comparison operations.	
[27]	QC	Cumulative saturation bit, Advanced SIMD only. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated since 0 was last written to this bit.	
[26]	AHP	Alternative half-precision control bit: **`0b0`** IEEE half-precision format selected. **`0b1`** Alternative half-precision format selected. This bit is only used for conversions between half-precision floating-point and other floating-point formats. The data-processing instructions added as part of the ARMv8.2-FP16 extension always use the IEEE half-precision format, and ignore the value of this bit.	
[25]	DN	Default NaN mode control bit: **`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. The value of this bit only controls scalar floating-point arithmetic. Advanced SIMD arithmetic always uses the Default NaN setting, regardless of the value of the DN bit.	
[24]	FZ	Flush-to-zero mode control bit: **`0b0`** Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. **`0b1`** Flush-to-zero mode enabled. The value of this bit only controls scalar floating-point arithmetic. Advanced SIMD arithmetic always uses the Flush-to-zero setting, regardless of the value of the FZ bit. This bit has no effect on half-precision calculations.	

Bits	Name	Description	Reset
[23:22]	RMode	Rounding Mode control field. The encoding of this field is: **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 almost all scalar floating-point instructions. Advanced SIMD arithmetic always uses the Round to Nearest setting, regardless of the value of the RMode bits.	
[21:20]	RES0	Reserved	0b00
[19]	FZ16	Flush-to-zero mode control bit on half-precision data-processing instructions: **0b0** Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. **0b1** Flush-to-zero mode enabled. The value of this bit applies to both scalar and Advanced SIMD floating-point half-precision calculations.	
[18:8]	RES0	Reserved	0b0
[7]	IDC	Input Denormal cumulative floating-point exception bit. This bit is set to 1 to indicate that the Input Denormal floating-point exception has occurred since 0 was last written to this bit. How VFP instructions update this bit depends on the value of the IDE bit. Advanced SIMD instructions set this bit if the Input Denormal floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the IDE bit.	
[6:5]	RES0	Reserved	0b00
[4]	IXC	Inexact cumulative floating-point exception bit. This bit is set to 1 to indicate that the Inexact floating-point exception has occurred since 0 was last written to this bit. How VFP instructions update this bit depends on the value of the IXE bit. Advanced SIMD instructions set this bit if the Inexact floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the IXE bit. The criteria for the Inexact floating-point exception to occur are different in Flush-to-zero mode. For details, see 'Flush-to-zero'.	

Bits	Name	Description	Reset
[3]	UFC	Underflow cumulative floating-point exception bit. This bit is set to 1 to indicate that the Underflow floating-point exception has occurred since 0 was last written to this bit. How VFP instructions update this bit depends on the value of the UFE bit. Advanced SIMD instructions set this bit if the Underflow floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the UFE bit. The criteria for the Underflow floating-point exception to occur are different in Flush-to-zero mode. For details, see 'Flush-to-zero'.	
[2]	OFC	Overflow cumulative floating-point exception bit. This bit is set to 1 to indicate that the Overflow floating-point exception has occurred since 0 was last written to this bit. How VFP instructions update this bit depends on the value of the OFE bit. Advanced SIMD instructions set this bit if the Overflow floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the OFE bit.	
[1]	DZC	Divide by Zero cumulative floating-point exception bit. This bit is set to 1 to indicate that the Divide by Zero floating-point exception has occurred since 0 was last written to this bit. How VFP instructions update this bit depends on the value of the DZE bit. Advanced SIMD instructions set this bit if the Divide by Zero floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the DZE bit.	
[0]	IOC	Invalid Operation cumulative floating-point exception bit. This bit is set to 1 to indicate that the Invalid Operation floating-point exception has occurred since 0 was last written to this bit. How VFP instructions update this bit depends on the value of the IOE bit. Advanced SIMD instructions set this bit if the Invalid Operation floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the IOE bit.	

Appendix B AArch64 registers

This appendix contains the descriptions for the Neoverse™ N2 AArch64 registers.

B.1 AArch64 generic system control registers

The summary table provides an overview of all implementation defined generic system control registers in the core. Individual register descriptions provide detailed information.

Table B-1: generic system control register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
AIDR_EL1	3	C0	1	C0	7	`0x0`	64-bit	Auxiliary ID Register
ACTLR_EL1	3	C1	0	C0	1	`0x0`	64-bit	Auxiliary Control Register (EL1)
ACTLR_EL2	3	C1	4	C0	1	`0x0`	64-bit	Auxiliary Control Register (EL2)
HACR_EL2	3	C1	4	C1	7	`0x0`	64-bit	Hypervisor Auxiliary Control Register
ACTLR_EL3	3	C1	6	C0	1	`0x0`	64-bit	Auxiliary Control Register (EL3)
AMAIR_EL2	3	C10	0	C3	0	`0x0`	64-bit	Auxiliary Memory Attribute Indirection Register (EL2)
LORID_EL1	3	C10	0	C4	7	See individual bit resets.	64-bit	LORegionID (EL1)
AMAIR_EL1	3	C10	5	C3	0	`0x0`	64-bit	Auxiliary Memory Attribute Indirection Register (EL1)
AMAIR_EL3	3	C10	6	C3	0	`0x0`	64-bit	Auxiliary Memory Attribute Indirection Register (EL3)
RMR_EL3	3	C12	6	C0	2	See individual bit resets.	64-bit	Reset Management Register (EL3)
IMP_CPUACTLR_EL1	3	C15	0	C1	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register (EL1)
IMP_CPUACTLR2_EL1	3	C15	0	C1	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 2 (EL1)
IMP_CPUACTLR3_EL1	3	C15	0	C1	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 3 (EL1)
IMP_CPUACTLR4_EL1	3	C15	0	C1	3	See individual bit resets.	64-bit	CPU Auxiliary Control Register 4 (EL1)
IMP_CPUECTLR_EL1	3	C15	0	C1	4	See individual bit resets.	64-bit	CPU Extended Control Register
IMP_CPUECTLR2_EL1	3	C15	0	C1	5	See individual bit resets.	64-bit	CPU Extended Control Register 2
IMP_CPUPPMCR3_EL3	3	C15	0	C2	4	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPWRCTLR_EL1	3	C15	0	C2	7	`0x0`	64-bit	CPU Power Control Register
IMP_ATCR_EL1	3	C15	0	C7	0	`0x0`	64-bit	CPU Auxiliary Translation Control Register (EL1)
IMP_CPUACTLR5_EL1	3	C15	0	C8	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register 5 (EL1)

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
IMP_CPUACTLR6_EL1	3	C15	0	C8	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 6 (EL1)
IMP_CPUACTLR7_EL1	3	C15	0	C8	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 7 (EL1)
IMP_ATCR_EL2	3	C15	4	C7	0	`0x0`	64-bit	CPU Auxiliary Translation Control Register (EL2)
IMP_AVTCR_EL2	3	C15	4	C7	1	`0x0`	64-bit	CPU Virtualization Auxiliary Translation Control Register (EL2)
IMP_CPUPPMCR_EL3	3	C15	6	C2	0	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR2_EL3	3	C15	6	C2	1	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR4_EL3	3	C15	6	C2	4	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR5_EL3	3	C15	6	C2	5	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR6_EL3	3	C15	6	C2	6	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUACTLR_EL3	3	C15	6	C4	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register (EL3)
IMP_ATCR_EL3	3	C15	6	C7	0	`0x0`	64-bit	CPU Auxiliary Translation Control Register (EL2)
IMP_CPUPSELR_EL3	3	C15	6	C8	0	See individual bit resets.	64-bit	Selected Instruction Private Select Register
IMP_CPUPCR_EL3	3	C15	6	C8	1	See individual bit resets.	64-bit	Selected Instruction Private Control Register
IMP_CPUPOR_EL3	3	C15	6	C8	2	See individual bit resets.	64-bit	Selected Instruction Private Opcode Register
IMP_CPUPMR_EL3	3	C15	6	C8	3	See individual bit resets.	64-bit	Selected Instruction Private Mask Register
IMP_CPUPOR2_EL3	3	C15	6	C8	4	See individual bit resets.	64-bit	Selected Instruction Private Opcode Register 2
IMP_CPUPMR2_EL3	3	C15	6	C8	5	See individual bit resets.	64-bit	Selected Instruction Private Mask Register 2
IMP_CPUPFR_EL3	3	C15	6	C8	6	See individual bit resets.	64-bit	Selected Instruction Private Flag Register
FPCR	3	C4	3	C4	0	See individual bit resets.	64-bit	Floating-point Control Register
AFSR0_EL2	3	C5	0	C1	0	`0x0`	64-bit	Auxiliary Fault Status Register 0 (EL2)
AFSR1_EL2	3	C5	0	C1	1	`0x0`	64-bit	Auxiliary Fault Status Register 1 (EL2)
AFSR0_EL1	3	C5	5	C1	0	`0x0`	64-bit	Auxiliary Fault Status Register 0 (EL1)
AFSR1_EL1	3	C5	5	C1	1	`0x0`	64-bit	Auxiliary Fault Status Register 1 (EL1)
AFSR0_EL3	3	C5	6	C1	0	`0x0`	64-bit	Auxiliary Fault Status Register 0 (EL3)
AFSR1_EL3	3	C5	6	C1	1	`0x0`	64-bit	Auxiliary Fault Status Register 1 (EL3)

B.1.1 AIDR_EL1, Auxiliary ID Register

Provides **IMPLEMENTATION DEFINED** identification information.

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

`0x0`

Bit descriptions

Figure B-1: AArch64_aidr_el1 bit assignments

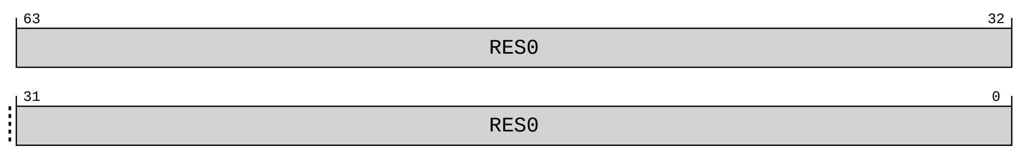


Table B-2: AIDR_EL1 bit descriptions

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

Access

MRS <Xt>, AIDR_EL1

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

Accessibility

MRS <Xt>, AIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AIDR_EL1;
elsif PSTATE.EL == EL2 then
    return AIDR_EL1;
elsif PSTATE.EL == EL3 then
    return AIDR_EL1;
```

B.1.2 ACTLR_EL1, Auxiliary Control Register (EL1)

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

Note

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

0x0

Bit descriptions

Figure B-2: AArch64_actlr_el1 bit assignments

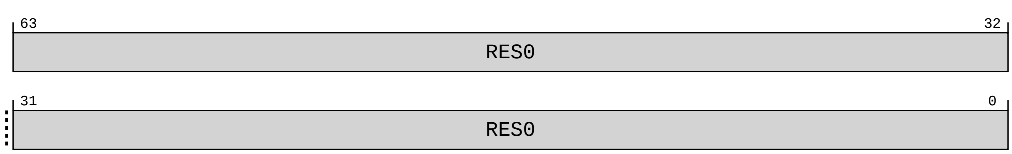


Table B-4: ACTLR_EL1 bit descriptions

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

Access

MRS <Xt>, ACTLR_EL1

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

MSR ACTLR_EL1, <Xt>

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

Accessibility

MRS <Xt>, ACTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        return NVMem[0x118];
    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);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        NVMem[0x118] = X[t];
    else
        ACTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    ACTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ACTLR_EL1 = X[t];
```

B.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

Reset value

`0x0`

Bit descriptions

Figure B-3: AArch64_actlr_el2 bit assignments

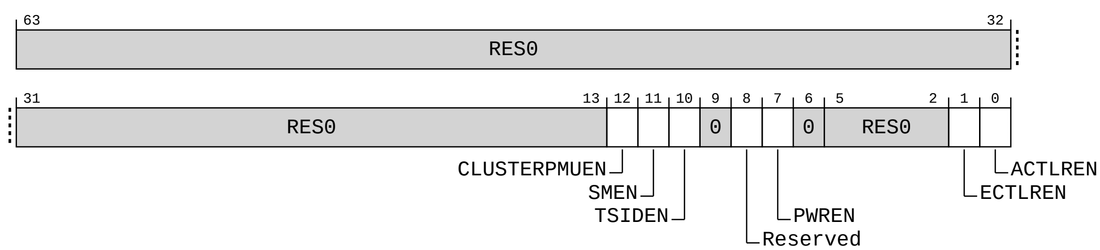


Table B-7: ACTLR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:13]	RES0	Reserved	`0x0`
[12]	CLUSTERPMUEN	Performance Management Registers enable. The possible values are: **0b0** CLUSTERPM* registers are not write-accessible from EL1. This is the reset value. **0b1** CLUSTERPM* registers are write-accessible from EL1 if they are write-accessible from EL2.	`0x0`
[11]	SMEN	Scheme Management Registers enable. The possible values are: **0b0** Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible EL1. This is the reset value. **0b1** Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible EL1 if they are write-accessible from EL2.	`0x0`
[10]	TSIDEN	Thread Scheme ID Register enable. The possible values are: **0b0** Register CLUSTERTHREADSID is not write-accessible from EL1. This is the reset value. **0b1** Register CLUSTERTHREADSID is write-accessible from EL1 if they are write-accessible from EL2	`0x0`

Bits	Name	Description	Reset
[9]	RES0	Reserved	0x0
[8]	Reserved	Reserved **0b0** Reserved **0b1** Reserved	0x0
[7]	PWREN	Power Control Registers enable. The possible values are: **0b0** Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write accessible from EL1. This is the reset value. **0b1** Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible from EL1 if they are write-accessible from EL2	0x0
[6:2]	RES0	Reserved	0b0000
[1]	ECTLREN	Extended Control Registers enable. The possible values are: **0b0** CPUECTLR*_EL1 and CLUSTERECTLR_EL1 are not write-accessible from EL1. This is the reset value. **0b1** CPUECTLR*_EL1 and CLUSTERECTLR_EL1 are write-accessible from EL1 if they are write-accessible from EL2.	0b0
[0]	ACTLREN	Auxiliary Control Registers enable. The possible values are: **0b0** CPUACTLR*_EL1 and CLUSTERACTLR are not write-accessible from EL1. This is the reset value. **0b1** CPUACTLR*_EL1 and CLUSTERACTLR are write-accessible from EL1 if they are write-accessible from EL2	0b0

Access

MRS <Xt>, ACTLR_EL2

<systemreg>	op0	op1	CRn	CRm	op2
ACTLR_EL2	0b11	0b100	0b0001	0b0000	0b001

MSR ACTLR_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ACTLR_EL2	0b11	0b100	0b0001	0b0000	0b001

Accessibility

MRS <Xt>, ACTLR_EL2

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

B.1.4 HACR_EL2, Hypervisor Auxiliary Control Register

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

Note

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

Configurations

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

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

Attributes

Width

64

Functional group

generic-system-control

Reset value

`0x0`

Bit descriptions

Figure B-4: AArch64_hacr_el2 bit assignments

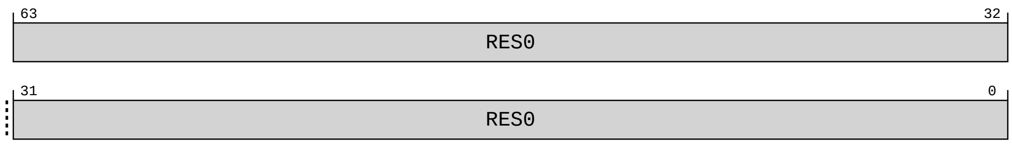


Table B-10: HACR_EL2 bit descriptions

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

Access

MRS <Xt>, HACR_EL2

<systemreg>	op0	op1	CRn	CRm	op2
HACR_EL2	`0b11`	`0b100`	`0b0001`	`0b0001`	`0b111`

MSR HACR_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
HACR_EL2	`0b11`	`0b100`	`0b0001`	`0b0001`	`0b111`

Accessibility

MRS <Xt>, HACR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return HACR_EL2;
elsif PSTATE.EL == EL3 then
    return HACR_EL2;
```

MSR HACR_EL2, <Xt>

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

```
    HACR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    HACR_EL2 = X[t];
```

B.1.5 ACTLR_EL3, Auxiliary Control Register (EL3)

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

`0x0`

Bit descriptions

Figure B-5: AArch64_actlr_el3 bit assignments

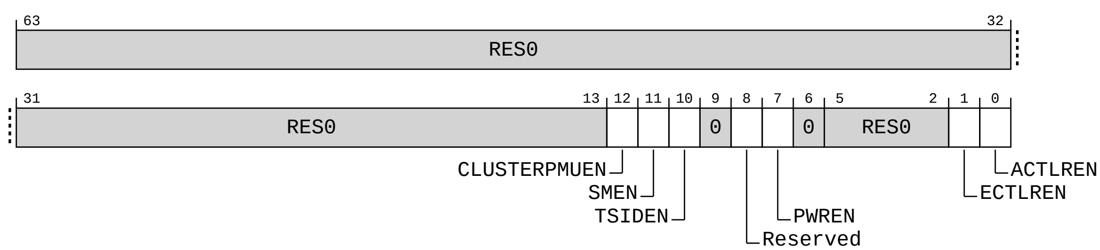


Table B-13: ACTLR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:13]	RES0	Reserved	`0x0`
[12]	CLUSTERPMUEN	Performance Management Registers enable. The possible values are: **`0b0`** CLUSTERPM* registers are not write-accessible from EL2 and EL1. This is the reset value. **`0b1`** CLUSTERPM* registers are write-accessible from EL2 and EL1 if they are write-accessible from EL2.	`0x0`

Bits	Name	Description	Reset
[11]	SMEN	Scheme Management Registers enable. The possible values are: **0b0** Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible EL2 and EL1. This is the reset value. **0b1** Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible EL2 and EL1 if they are write-accessible from EL2.	0x0
[10]	TSIDEN	Thread Scheme ID Register enable. The possible values are: **0b0** Register CLUSTERTHREADSID is not write-accessible from EL2 and EL1. This is the reset value. **0b1** Register CLUSTERTHREADSID is write-accessible from EL2 and EL1 if they are write-accessible from EL2	0x0
[9]	RES0	Reserved	0x0
[8]	Reserved	Reserved **0b0** Reserved **0b1** Reserved	0x0
[7]	PWREN	Power Control Registers enable. The possible values are: **0b0** Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write accessible from EL2 and EL1. This is the reset value. **0b1** Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible from EL2 and EL1 if they are write-accessible from EL2	0x0
[6:2]	RES0	Reserved	0b0000
[1]	ECTLREN	Extended Control Registers enable. The possible values are: **0b0** CPUECTLR*_EL2 and EL1 and CLUSTERECTLR_EL2 and EL1 are not write-accessible from EL2 and EL1. This is the reset value. **0b1** CPUECTLR*_EL2 and EL1 and CLUSTERECTLR_EL2 and EL1 are write-accessible from EL2 and EL1 if they are write-accessible from EL2.	0b0

Bits	Name	Description	Reset
[0]	ACTLREN	Auxiliary Control Registers enable. The possible values are: **0b0** CPUACTLR*_EL2 and EL1 and CLUSTERACTLR are not write-accessible from EL2 and EL1. This is the reset value. **0b1** CPUACTLR*_EL2 and EL1 and CLUSTERACTLR are write-accessible from EL2 and EL1 if they are write-accessible from EL2	`0b0`

Access

MRS <Xt>, ACTLR_EL3

<systemreg>	op0	op1	CRn	CRm	op2
ACTLR_EL3	`0b11`	`0b110`	`0b0001`	`0b0000`	`0b001`

MSR ACTLR_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ACTLR_EL3	`0b11`	`0b110`	`0b0001`	`0b0000`	`0b001`

Accessibility

MRS <Xt>, ACTLR_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
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];
```

B.1.6 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2)

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

Configurations

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

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

Attributes

Width

64

Functional group

generic-system-control

Reset value

`0x0`

Bit descriptions

AMAIR_EL2 is permitted to be cached in a TLB.

Figure B-6: AArch64_amair_el2 bit assignments

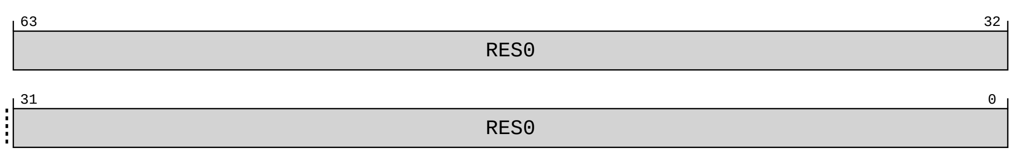


Table B-16: AMAIR_EL2 bit descriptions

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

Access

MRS <Xt>, AMAIR_EL2

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL2	0b11	0b100	0b1010	0b0011	0b000

MSR AMAIR_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL2	0b11	0b100	0b1010	0b0011	0b000

MRS <Xt>, AMAIR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	0b11	0b000	0b1010	0b0011	0b000

MSR AMAIR_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	0b11	0b000	0b1010	0b0011	0b000

Accessibility

MRS <Xt>, AMAIR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        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
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AMAIR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    AMAIR_EL2 = X[t];
```

MRS <Xt>, AMAIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x148];
    else
        return AMAIR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AMAIR_EL2;
    else
        return AMAIR_EL1;
elsif PSTATE.EL == EL3 then
    return AMAIR_EL1;
```

MSR AMAIR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x148] = X[t];
    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];
```

B.1.7 LORID_EL1, LORegionID (EL1)

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

Configurations

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

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-7: AArch64_lorid_el1 bit assignments

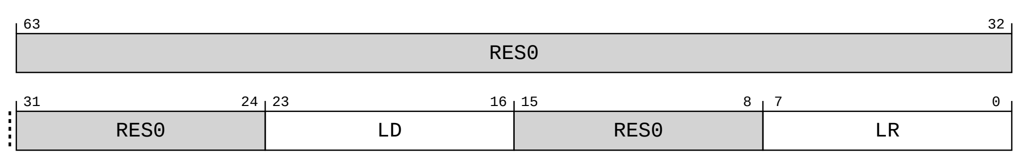


Table B-21: LORID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[23:16]	LD	Number of LORegion descriptors supported by the PE. This is an 8-bit binary number. **0b00000100** Four LOR descriptors are supported	
[15:8]	RES0	Reserved	0b00000000
[7:0]	LR	Number of LORegions supported by the PE. This is an 8-bit binary number. **Note:** If LORID_EL1 indicates that no LORegions are implemented, then LoadLOAcquire and StoreLORelease will behave as LoadAcquire and StoreRelease. **0b00000100** Four LORegions are supported	

Access

MRS <Xt>, LORID_EL1

<systemreg>	op0	op1	CRn	CRm	op2
LORID_EL1	0b11	0b000	0b1010	0b0100	0b111

Accessibility

MRS <Xt>, LORID_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.LORID_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TLOR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return LORID_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TLOR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return LORID_EL1;
elsif PSTATE.EL == EL3 then
    return LORID_EL1;
```

B.1.8 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register (EL1)

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

0x0

Bit descriptions

AMAIR_EL1 is permitted to be cached in a TLB.

Figure B-8: AArch64_amair_el1 bit assignments

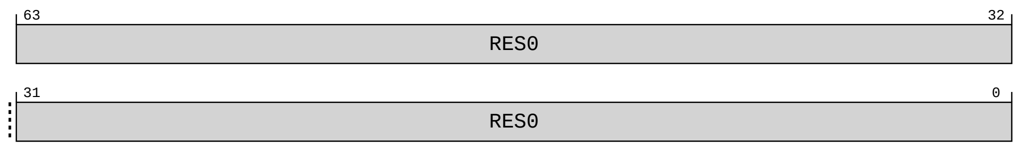


Table B-23: AMAIR_EL1 bit descriptions

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

Access

MRS <Xt>, AMAIR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	0b11	0b000	0b1010	0b0011	0b000

MSR AMAIR_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	0b11	0b000	0b1010	0b0011	0b000

MRS <Xt>, AMAIR_EL12

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL12	0b11	0b101	0b1010	0b0011	0b000

MSR AMAIR_EL12, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL12	0b11	0b101	0b1010	0b0011	0b000

Accessibility

MRS <Xt>, AMAIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x148];
    else
        return AMAIR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AMAIR_EL2;
    else
        return AMAIR_EL1;
elsif PSTATE.EL == EL3 then
    return AMAIR_EL1;
```

MSR AMAIR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x148] = X[t];
    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
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        return NVMem[0x148];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return 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
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x148] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t];
    else
        UNDEFINED;
```

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

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

`0x0`

Bit descriptions

AMAIR_EL3 is permitted to be cached in a TLB.

Figure B-9: AArch64_amair_el3 bit assignments

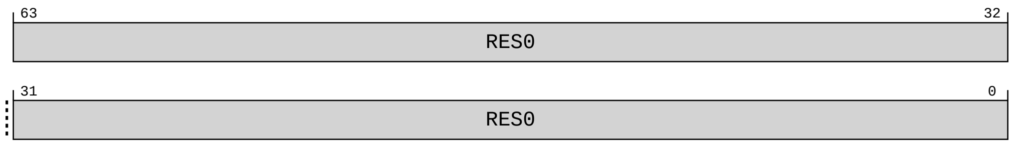


Table B-28: AMAIR_EL3 bit descriptions

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

Access

MRS <Xt>, AMAIR_EL3

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL3	0b11	0b110	0b1010	0b0011	0b000

MSR AMAIR_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL3	0b11	0b110	0b1010	0b0011	0b000

Accessibility

MRS <Xt>, AMAIR_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return AMAIR_EL3;
```

MSR AMAIR_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AMAIR_EL3 = X[t];
```

B.1.10 RMR_EL3, Reset Management Register (EL3)

A write to the register at EL3 can request a Warm reset.

Configurations

When EL3 is implemented:

- If EL3 can use all Execution states then this register must be implemented.
- If EL3 cannot use AArch32, then it is IMPLEMENTATION DEFINED whether the register is implemented.

Otherwise, direct accesses to RMR_EL3 are UNDEFINED.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-10: AArch64_rmr_el3 bit assignments

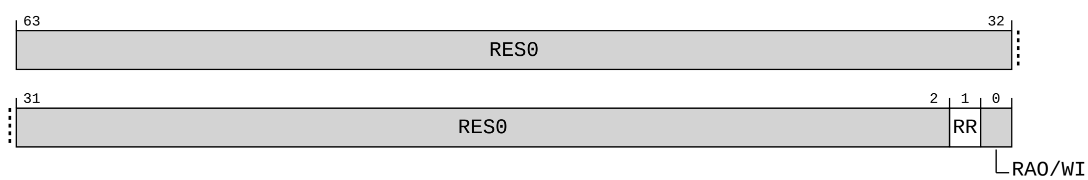


Table B-31: RMR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:2]	RES0	Reserved	`0x0`
[1]	RR	Reset Request. Setting this bit to 1 requests a Warm reset.	`0x0`
[0]	RAO/WI	Reserved	

Access

MRS <Xt>, RMR_EL3

<systemreg>	op0	op1	CRn	CRm	op2
RMR_EL3	`0b11`	`0b110`	`0b1100`	`0b0000`	`0b010`

MSR RMR_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
RMR_EL3	0b11	0b110	0b1100	0b0000	0b010

Accessibility

MRS <Xt>, RMR_EL3

```
if PSTATE.EL == EL3 then
    return RMR_EL3;
else
    UNDEFINED;
```

MSR RMR_EL3, <Xt>

```
if PSTATE.EL == EL3 then
    RMR_EL3 = X[t];
else
    UNDEFINED;
```

B.1.11 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-11: AArch64_imp_cpuactlr_el1 bit assignments

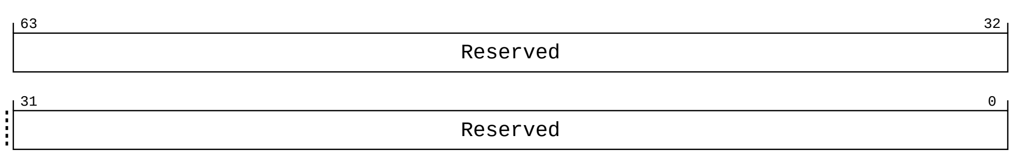


Table B-34: IMP_CPUACTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_0_C15_C1_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_0	0b11	0b000	0b1111	0b0001	0b000

MSR S3_0_C15_C1_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_0	0b11	0b000	0b1111	0b0001	0b000

Accessibility

MRS <Xt>, S3_0_C15_C1_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUACTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUACTLR_EL1;
```

MSR S3_0_C15_C1_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR_EL1 = X[t];
```

B.1.12 IMP_CPUACTLR2_EL1, CPU Auxiliary Control Register 2 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-12: AArch64_imp_cpuactlr2_el1 bit assignments

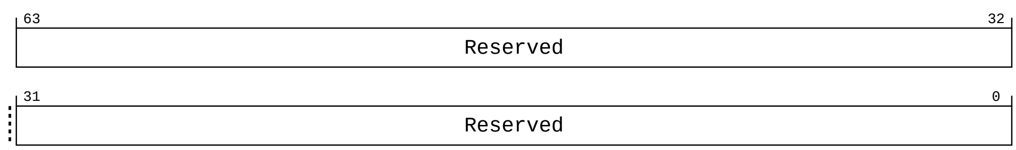


Table B-37: IMP_CPUACTLR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_0_C15_C1_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_1	0b11	0b000	0b1111	0b0001	0b001

MSR S3_0_C15_C1_1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_1	0b11	0b000	0b1111	0b0001	0b001

Accessibility

MRS <Xt>, S3_0_C15_C1_1

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

```
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR2_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUACTLR2_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUACTLR2_EL1;
```

MSR S3_0_C15_C1_1, <Xt>

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

B.1.13 IMP_CPUACTLR3_EL1, CPU Auxiliary Control Register 3 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-13: AArch64_imp_cpuactlr3_el1 bit assignments

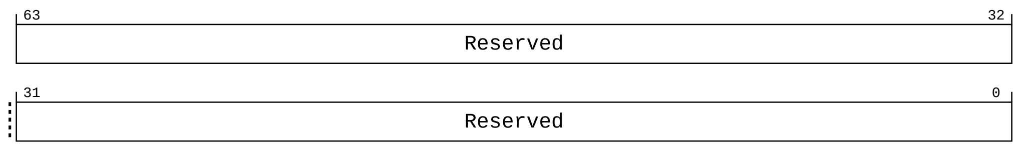


Table B-40: IMP_CPUACTLR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_0_C15_C1_2

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_2	0b11	0b000	0b1111	0b0001	0b010

MSR S3_0_C15_C1_2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_2	0b11	0b000	0b1111	0b0001	0b010

Accessibility

MRS <Xt>, S3_0_C15_C1_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR3_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUACTLR3_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUACTLR3_EL1;
```

MSR S3_0_C15_C1_2, <Xt>

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

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

B.1.14 IMP_CPUACTLR4_EL1, CPU Auxiliary Control Register 4 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-14: AArch64_imp_cpuactlr4_el1 bit assignments

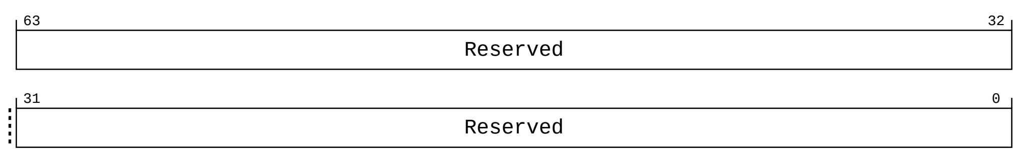


Table B-43: IMP_CPUACTLR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_0_C15_C1_3

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_3	0b11	0b000	0b1111	0b0001	0b011

MSR S3_0_C15_C1_3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_3	0b11	0b000	0b1111	0b0001	0b011

Accessibility

MRS <Xt>, S3_0_C15_C1_3

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

MSR S3_0_C15_C1_3, <Xt>

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

B.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

Reset value

See individual bit resets.

Bit descriptions

Figure B-15: AArch64_imp_cpuectlr_el1 bit assignments

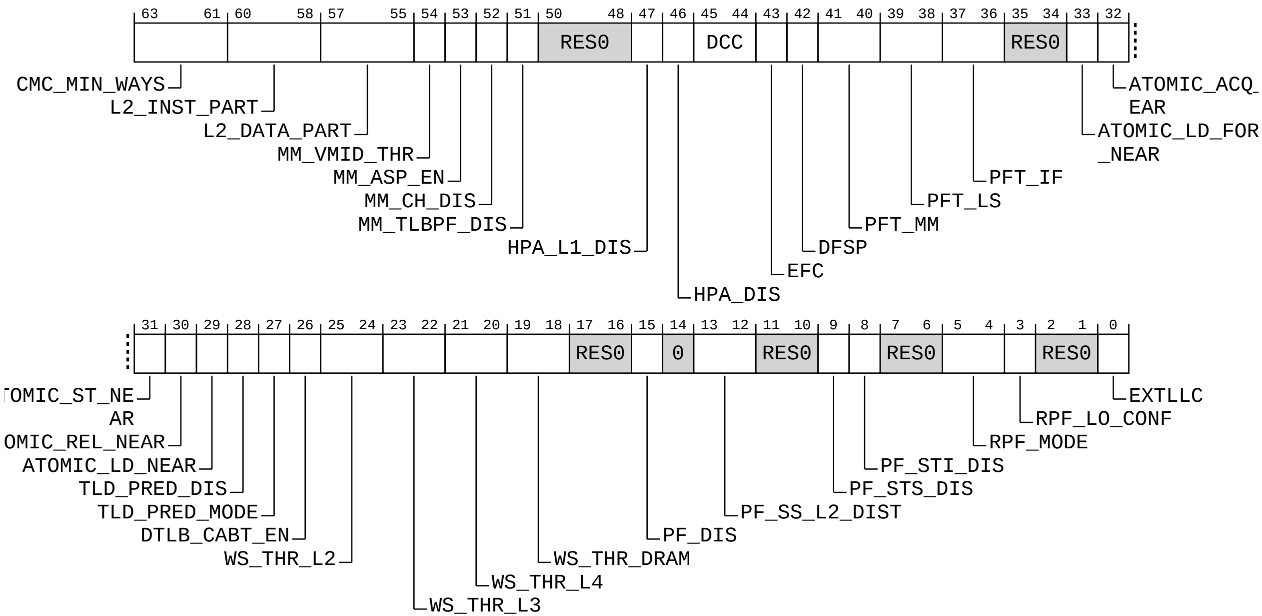


Table B-46: IMP_CPUECTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:61]	CMC_MIN_WAYS	Limits how many ways of L2 can be used by CMC. The possible values are: **`0b000`** CMC disabled **`0b001`** CMC must leave at least 1 way for data in L2 **`0b010`** CMC must leave at least 2 ways for data in L2 - This is the default value. **`0b011`** CMC must leave at least 3 ways for data in L2 **`0b100`** CMC must leave at least 4 ways for data in L2 **`0b101`** CMC must leave at least 5 ways for data in L2 **`0b110`** CMC must leave at least 6 ways for data in L2 **`0b111`** CMC must leave at least 7 ways for data in L2	
[60:58]	L2_INST_PART	Partition the L2 cache for Instruction. The possible values are: **`0b000`** No ways reserved for instructions. This is the reset value **`0b001`** Reserve 1 way for instruction. Only instruction fetches can allocate way 7 **`0b010`** Reserve 2 ways for instruction. Only instruction fetches can allocate ways 7:6 **`0b011`** Reserve 3 ways for instruction. Only instruction fetches can allocate ways 7:5 **`0b100`** Reserve 4 ways for instruction. Only instruction fetches can allocate ways 7:4 **`0b101`** Reserve 5 ways for instruction. Only instruction fetches can allocate ways 7:3 **`0b110`** Reserve 6 ways for instruction. Only instruction fetches can allocate ways 7:2 **`0b111`** Reserve 7 ways for instruction. Only instruction fetches can allocate ways 7:1	

Bits	Name	Description	Reset

[57:55]	L2_DATA_PART	Reserve L2 capacity for data accesses. The possible values are:
		`0b000`
		No ways reserved for data. This is the reset value
		`0b001`
		Reserve 1 way for data. Only data accesses can allocate way 0
		`0b010`
		Reserve 2 ways for data. Only data accesses can allocate ways 1:0
		`0b011`
		Reserve 3 ways for data. Only data accesses can allocate ways 2:0
		`0b100`
		Reserve 4 ways for data. Only data accesses can allocate ways 3:0
		`0b101`
		Reserve 5 ways for data. Only data accesses can allocate ways 4:0
		`0b110`
		Reserve 6 ways for data. Only data accesses can allocate ways 5:0
		`0b111`
		Reserve 7 ways for data. Only data accesses can allocate ways 6:0	

[54]	MM_VMID_THR	VMID filter threshold. The possible values are:
		`0b0`
		VMID filter flush after 16 unique VMID allocations to the MMU Translation Cache. This is the default value.
		`0b1`
		VMID filter flush after 32 unique VMID allocations to the MMU Translation Cache	

[53]	MM_ASP_EN	Disables allocation of splintered pages in L2 TLB. The possible values are:
		`0b0`
		Enables allocation of splintered pages in the L2 TLB. This is the default value.
		`0b1`
		Disables allocation of splintered pages in the L2 TLB.	

[52]	MM_CH_DIS	Disables use of contiguous hint. The possible values are:
		`0b0`
		Enables use of contiguous hint. This is the default value.
		`0b1`
		Disables use of contiguous hint.	

Bits	Name	Description	Reset
[51]	MM_TLBPF_DIS	Disables TLB prefetcher. The possible values are: **0b0** Enables TLB prefetcher. This is the default value. **0b1** Disables TLB prefetcher.	
[50:48]	RES0	Reserved	0b000
[47]	HPA_L1_DIS	Disables hardware page aggregation in L1 TLBs. The possible values are: **0b0** Enables hardware page aggregation in L1 TLBs. This is the default value. **0b1** Disables hardware page aggregation in L1 TLBs.	
[46]	HPA_DIS	Disable Hardware page aggregation. The possible values are: **0b0** Enables hardware page aggregation. This is the default value. **0b1** Disables hardware page aggregation.	
[45:44]	DCC	Downstream Cache Control. Controls whether evictions of clean cache-lines send data on the CHI interface. Set this based on whether there is a cache on the path to memory. The possible values are: **0b00** Disables sending data when clean cache-lines are evicted. **0b01** Enables sending WriteEvictFull transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data. **0b10** Enables sending WriteEvictOrEvict transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data. This is the default value when the SCU is not present **0b11** Enables sending WriteEvictOrEvict transactions when Unique Clean or Shared Clean cache-lines are evicted. This is the default value when SCU is present	
[43]	EFC	Eviction flush control. Controls whether hardware cache flushes and DC CISW instructions send data when evicting clean cachelines on the CHI interface. The possible values are: **0b0** Disables sending data when hardware cache flushes or DC CISW instructions evict a clean cacheline. Sending of Evict transactions is controlled by Downstream Snoop Filter Present (DSFP). This is the default value. **0b1** Sending of data when hardware cache flushes or DC CISW instructions evict clean cachelines is controlled by Downstream Cache Control (DCC). Sending of Evict transactions is controlled by Downstream Snoop Filter Present (DSFP).	

Bits	Name	Description	Reset
[42]	DFSP	Downstream snoop filter present. Enables sending Evict transactions on the CHI when clean cachelines are evicted without data. Enable this if there is at least one snoop filter in the path to memory. The possible values are: **`0b0`** Disables sending Evict transactions when clean cachelines are evicted without data **`0b1`** Enables sending of Evict transactions when clean cachelines are evicted without data. This is the default value	
[41:40]	PFT_MM	DRAM prefetch using PrefetchTgt transactions for tablewalk requests. The possible values are: **`0b00`** Disable prefetchtgt generation for requests from the Memory Management unit (MMU). This is the default value. **`0b01`** Conservatively generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable. **`0b10`** Agressively generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable. **`0b11`** Always generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable.	
[39:38]	PFT_LS	DRAM prefetch using PrefetchTgt transactions for load and store requests. The possible values are: **`0b00`** Disable prefetchtgt generation for requests from the Load-Store unit (LS). This is the default value. **`0b01`** Conservatively generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable. **`0b10`** Agressively generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable. **`0b11`** Always generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable.	

Bits	Name	Description	Reset
[37:36]	PFT_IF	DRAM prefetch using PrefetchTgt transactions for instruction fetch requests. The possible values are: **`0b00`** Disable prefetchtgt generation for requests from the Instruction Fetch unit (IF). This is the default value. **`0b01`** Conservatively generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable. **`0b10`** Agressively generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable. **`0b11`** Always generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable.	
[35:34]	RES0	Reserved	`0b00`
[33]	ATOMIC_LD_FORCE_NEAR	A load atomic (including SWP & CAS) instruction to WB memory will be performed near. The possible values are: **`0b0`** Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request. **`0b1`** Load-atomic will be performed near by bringing the line into the L1D Cache. This is the default value.	
[32]	ATOMIC_ACQ_NEAR	An atomic instruction to WB memory with acquire semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **`0b0`** Acquire-atomic is near if cache line is already Exclusive, otherwise make far atomic request. **`0b1`** Acquire-atomic will make up to 1 fill request to perform near. This is the default value.	
[31]	ATOMIC_ST_NEAR	A store atomic instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **`0b0`** Store-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the default value. **`0b1`** Store-atomic will make up to 1 fill request to perform near.	

Bits	Name	Description	Reset
[30]	ATOMIC_REL_NEAR	An atomic instruction to WB memory with release semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **`0b0`** Release-atomic is near if cache line is already Exclusive, otherwise make far atomic request. **`0b1`** Release-atomic will make up to 1 fill request to perform near. This is the default value.	
[29]	ATOMIC_LD_NEAR	A load atomic (including SWP & CAS) instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **`0b0`** Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the default value. **`0b1`** Load-atomic will make up to 1 fill request to perform near.	
[28]	TLD_PRED_DIS	Disable Transient Load Prediction. The possible values are: **`0b0`** Enables transient load prediction. This is the default value. **`0b1`** Disables transient load prediction.	
[27]	TLD_PRED_MODE	Aggressive Transient Load Prediction. The possible values are: **`0b0`** Disables aggressive transient load prediction. This is the default value. **`0b1`** Enables aggressive transient load prediction.	
[26]	DTLB_CABT_EN	Enables TLB Conflict Data Abort Exception. The possible values are: **`0b0`** Disables TLB conflict data abort exception. This is the default value. **`0b1`** Enables TLB conflict data abort exception.	
[25:24]	WS_THR_L2	Threshold for direct stream to L2 cache on store. The possible values are: **`0b00`** 256B - This is the default value **`0b01`** 4KB **`0b10`** 8KB **`0b11`** Disables direct stream to L2 cache on store.	

Bits	Name	Description	Reset
[23:22]	WS_THR_L3	Threshold for direct stream to L3 cache on store. The possible values are: **0b00** 128KB **0b01** 256KB - This is the default value **0b10** 512KB **0b11** Disables direct stream to L3 cache on store.	
[21:20]	WS_THR_L4	Threshold for direct stream to L4 cache on store. The possible values are: **0b00** 256KB **0b01** 512KB - This is the default value **0b10** 1MB **0b11** Disables direct stream to L4 cache on store.	
[19:18]	WS_THR_DRAM	Threshold for direct stream to DRAM on store. The possible values are: **0b00** 512KB **0b01** 1MB - This is the default value **0b10** 2MB **0b11** Disables direct stream to DRAM on store.	
[17:16]	RES0	Reserved	0b00
[15]	PF_DIS	Disables hardware prefetching. The possible values are: **0b0** Enables hardware prefetching. This is the default value. **0b1** Disables hardware prefetching.	
[14]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[13:12]	PF_SS_L2_DIST	Single cache line stride prefetching L2 distance. The possible values are: **0b00** 22 lines ahead **0b01** 40 lines ahead **0b10** 60 lines ahead **0b11** Dynamic. This is the default value.	
[11:10]	RES0	Reserved	0b00
[9]	PF_STS_DIS	Disable store-stride prefetches. The possible values are: **0b0** Enables store prefetching. This is the default value. **0b1** Disables store prefetching.	
[8]	PF_STI_DIS	Disable store prefetches at issue (not overridden by ls_hw_pref_disable). The possible values are: **0b0** Enables store prefetching. This is the default value. **0b1** Disable store prefetching.	
[7:6]	RES0	Reserved	0b00
[5:4]	RPF_MODE	Region prefetcher aggressiveness. The possible values are: **0b00** Dynamic region prefetch aggressiveness. This is the default value. **0b01** Conservative region prefetching. **0b10** Very Conservative region prefetching. **0b11** Most Conservative region prefetching. This will disable the region prefetcher.	
[3]	RPF_LO_CONF	Region Prefetcher single accesses training behavior. The possible values are: **0b0** Mostly don't train PHT on single access. This is the default value. **0b1** Always train the PHT on single access. This results in fewer prefetch requests.	
[2:1]	RES0	Reserved	0b00

Bits	Name	Description	Reset
[0]	EXTLLC	Internal or external Last-level cache (LLC) in the system. The possible values are: **0b0** Indicates that an internal Last-level cache is present in the system, and that the DataSource field on the master CHI interface indicates when data is returned from the LLC. This is used to control how the LL_CACHE* PMU events count. This is the default value. **0b1** Indicates that an external Last-level cache is present in the system, and that the DataSource field on the master CHI interface indicates when data is returned from the LLC. This is used to control how the LL_CACHE* PMU events count.	

Access

MRS <Xt>, S3_0_C15_C1_4

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_4	0b11	0b000	0b1111	0b0001	0b100

MSR S3_0_C15_C1_4, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_4	0b11	0b000	0b1111	0b0001	0b100

Accessibility

MRS <Xt>, S3_0_C15_C1_4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUECTLR_EL1;
```

MSR S3_0_C15_C1_4, <Xt>

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

```
    elsif ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUECTLR_EL1 = X[t];
```

B.1.16 IMP_CPUECTLR2_EL1, CPU Extended Control Register 2

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-16: AArch64_imp_cpuectlr2_el1 bit assignments

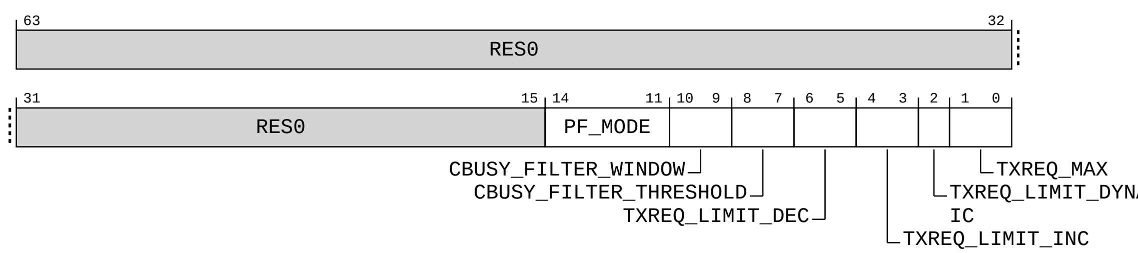


Table B-49: IMP_CPUECTLR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:15]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[14:11]	PF_MODE	Prefetcher Aggressiveness Modes. With mode 0 representing the most aggressive mode and 3 representing the most conservative mode. The possible values and associated ranges are: **`0b0000`** Modes [0,0] (statically at the most aggressive mode) **`0b0001`** Modes [0,1] **`0b0010`** Modes [0,2] **`0b0011`** Modes [0,3] - This is the default value. **`0b0100`** Modes [1,1] **`0b0101`** Modes [1,2] **`0b0110`** Modes [1,3] **`0b0111`** Modes [2,2] **`0b1000`** Modes [2,3] **`0b1001`** Modes [3,3] (statically at the most conservative mode) **`0b1010`** reserved **`0b1011`** reserved **`0b1100`** reserved **`0b1101`** reserved **`0b1110`** reserved **`0b1111`** reserved	

Bits	Name	Description	Reset
[10:9]	CBUSY_FILTER_WINDOW	Number of CBusy responses in one sampling window. The possible values are: **0b00** 256 - This is the default value **0b01** 64 **0b10** 128 **0b11** 512	
[8:7]	CBUSY_FILTER_THRESHOLD	Fraction of of CBusy responses in the sampling window necessary to be considered a valid sample of that CBusy value. The possible values are: **0b00** 1/16 - This is the default value **0b01** 1/32 **0b10** 1/8 **0b11** 1/4	
[6:5]	TXREQ_LIMIT_DEC	Dynamic TXREQ limit decrement. Controls how quickly the dynamic TXREQ limit is decreased when CBusy indicates value of 3. The possible values are: **0b00** 4 - This is the default value **0b01** 8 **0b10** 16 **0b11** 2	

Bits	Name	Description	Reset
[4:3]	TXREQ_LIMIT_INC	Dynamic TXREQ limit increment. Controls how quickly the dynamic TXREQ limit is increased when CBusy indicates values less than 2. The possible values are: **`0b00`** 4 - This is the default value **`0b01`** 8 **`0b10`** 16 **`0b11`** 2	
[2]	TXREQ_LIMIT_DYNAMIC	Selects static or dynamic control of TXREQ limit. Dynamic TXREQ limit will adjust based on CBusy responses on RXDAT and RXRSP in the range of the static limit selected by CPUECTLR2_EL1[1:0] and 1/4 of the L2 TQ SIZE. The possible values are: **`0b0`** maximum number of TXREQ transactions statically set by CPUECTLR2_EL1[1:0] - This is the default value. **`0b1`** maximum number of TXREQ transactions dynamically controlled	
[1:0]	TXREQ_MAX	Maximum number of TXREQ transactions outstanding from the L2 Transaction Queue. The possible values are: **`0b00`** full L2 TQ size - This is the default value **`0b01`** 3/4 of L2 TQ size **`0b10`** 1/2 of L2 TQ size **`0b11`** 1/4 of L2 TQ size	

Access

MRS <Xt>, S3_0_C15_C1_5

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_5	`0b11`	`0b000`	`0b1111`	`0b0001`	`0b101`

MSR S3_0_C15_C1_5, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_5	`0b11`	`0b000`	`0b1111`	`0b0001`	`0b101`

Accessibility

MRS <Xt>, S3_0_C15_C1_5

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

MSR S3_0_C15_C1_5, <Xt>

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

B.1.17 IMP_CPUPPMCR3_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-17: AArch64_imp_cpuppmcr3_el3 bit assignments

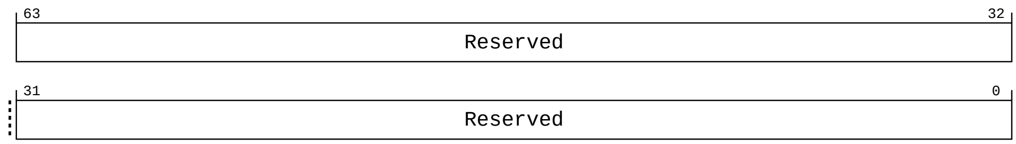


Table B-52: IMP_CPUPPMCR3_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_0_C15_C2_4

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_4	0b11	0b000	0b1111	0b0010	0b100

MSR S3_0_C15_C2_4, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_4	0b11	0b000	0b1111	0b0010	0b100

Accessibility

MRS <Xt>, S3_0_C15_C2_4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR3_EL3;
```

MSR S3_0_C15_C2_4, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR3_EL3 = X[t];
```

B.1.18 IMP_CPUPWRCTLR_EL1, CPU Power Control Register

This register controls various power aspects of the core.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

`0x0`

Bit descriptions

Figure B-18: AArch64_imp_cpupwrctlr_el1 bit assignments

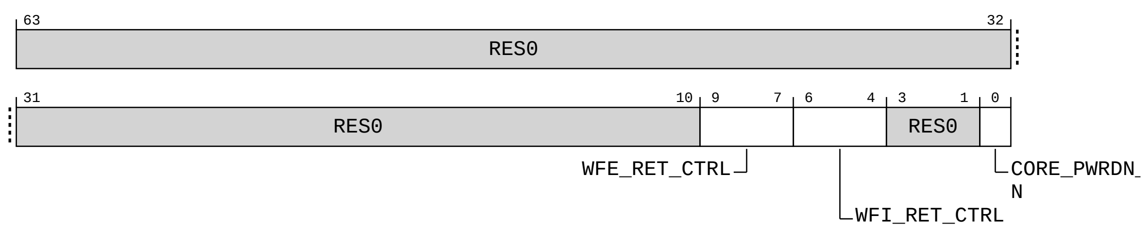


Table B-55: IMP_CPUPWRCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:10]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[9:7]	WFE_RET_CTRL	Wait for Event retention control. The possible values are: **0b000** Dynamic retention is disabled. **0b001** 2 system counter ticks are required before retention entry. **0b010** 8 system counter ticks are required before retention entry. **0b011** 32 system counter ticks are required before retention entry. **0b100** 64 system counter ticks are required before retention entry. **0b101** 128 system counter ticks are required before retention entry. **0b110** 256 system counter ticks are required before retention entry. **0b111** 512 system counter ticks are required before retention entry.	0x0
[6:4]	WFI_RET_CTRL	Wait for Interrupt retention control. The possible values are: **0b000** Dynamic retention is disabled. **0b001** 2 system counter ticks are required before retention entry. **0b010** 8 system counter ticks are required before retention entry. **0b011** 32 system counter ticks are required before retention entry. **0b100** 64 system counter ticks are required before retention entry. **0b101** 128 system counter ticks are required before retention entry. **0b110** 256 system counter ticks are required before retention entry. **0b111** 512 system counter ticks are required before retention entry.	0x0
[3:1]	RES0	Reserved	0b000

Bits	Name	Description	Reset
[0]	CORE_PWRDN_EN	Indicates to the power controller if the CPU wants to power down when it enters WFE/WFI state. The possible values are: **0b0** CPU does not want to power down when it enters WFE/WFI state. **0b1** CPU wants to power down when it enters WFE/WFI state.	0b0

Access

MRS <Xt>, S3_0_C15_C2_7

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_7	0b11	0b000	0b1111	0b0010	0b111

MSR S3_0_C15_C2_7, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_7	0b11	0b000	0b1111	0b0010	0b111

Accessibility

MRS <Xt>, S3_0_C15_C2_7

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUPWRCTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUPWRCTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPWRCTLR_EL1;
```

MSR S3_0_C15_C2_7, <Xt>

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

```
IMP_CPUPWRCTLR_EL1 = X[t];
```

B.1.19 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register (EL1)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

`0x0`

Bit descriptions

Figure B-19: AArch64_imp_atcr_el1 bit assignments

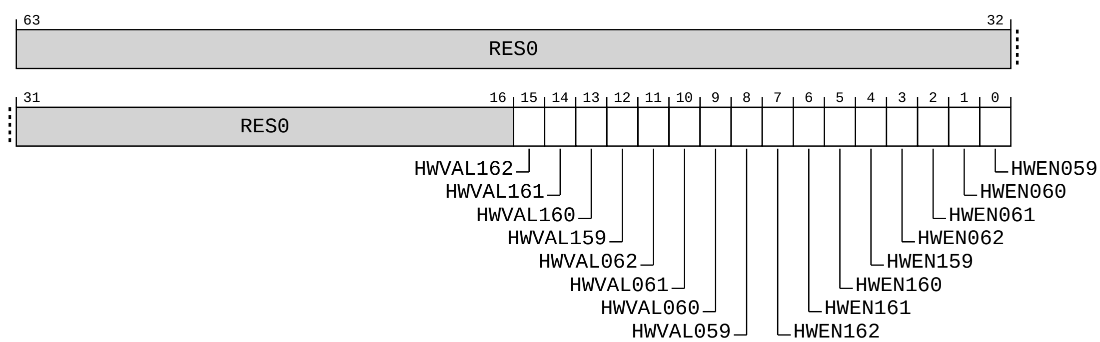


Table B-58: IMP_ATCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to page table walks using TTBR1_EL1 if HWEN162 is set.	0x0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to page table walks using TTBR1_EL1 if HWEN161 is set.	0x0
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to page table walks using TTBR1_EL1 if HWEN160 is set.	0x0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to page table walks using TTBR1_EL1 if HWEN159 is set.	0x0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using TTBR0_EL1 if HWEN062 is set.	0x0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using TTBR0_EL1 if HWEN061 is set.	0x0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using TTBR0_EL1 if HWEN060 is set.	0x0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using TTBR0_EL1 if HWEN059 is set.	0x0

Bits	Name	Description	Reset
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR1_EL1. If this bit is clear, PBHA[3] will be 0 on page table walks.	0x0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR1_EL1. If this bit is clear, PBHA[2] will be 0 on page table walks.	0x0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR1_EL1. If this bit is clear, PBHA[1] will be 0 on page table walks.	0x0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR1_EL1. If this bit is clear, PBHA[0] will be 0 on page table walks.	0x0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[3] will be 0 on page table walks.	0x0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[2] will be 0 on page table walks.	0x0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[1] will be 0 on page table walks.	0x0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[0] will be 0 on page table walks.	0x0

Access

MRS <Xt>, S3_0_C15_C7_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C7_0	0b11	0b000	0b1111	0b0111	0b000

MSR S3_0_C15_C7_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C7_0	0b11	0b000	0b1111	0b0111	0b000

Accessibility

MRS <Xt>, S3_0_C15_C7_0

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

MSR S3_0_C15_C7_0, <Xt>

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

```
    else
        IMP_ATCR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    IMP_ATCR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL1 = X[t];
```

B.1.20 IMP_CPUACTLR5_EL1, CPU Auxiliary Control Register 5 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-20: AArch64_imp_cpuactlr5_el1 bit assignments

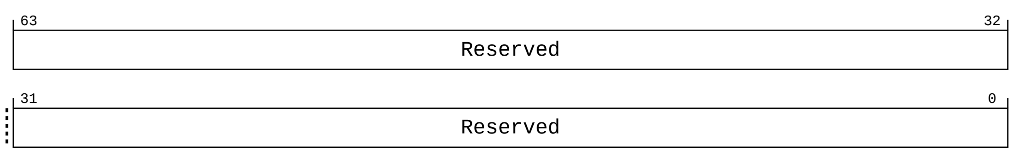


Table B-61: IMP_CPUACTLR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_0_C15_C8_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_0	0b11	0b000	0b1111	0b1000	0b000

MSR S3_0_C15_C8_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_0	0b11	0b000	0b1111	0b1000	0b000

Accessibility

MRS <Xt>, S3_0_C15_C8_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR5_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUACTLR5_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUACTLR5_EL1;
```

MSR S3_0_C15_C8_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR5_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR5_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR5_EL1 = X[t];
```

B.1.21 IMP_CPUACTLR6_EL1, CPU Auxiliary Control Register 6 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-21: AArch64_imp_cpuactlr6_el1 bit assignments

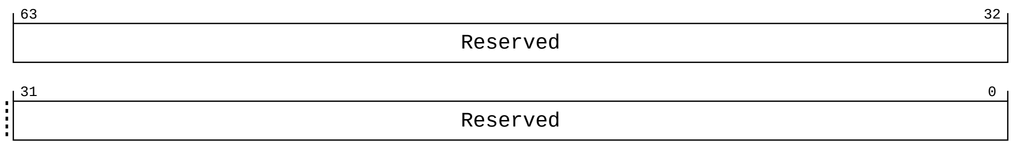


Table B-64: IMP_CPUACTLR6_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_0_C15_C8_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_1	0b11	0b000	0b1111	0b1000	0b001

MSR S3_0_C15_C8_1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_1	0b11	0b000	0b1111	0b1000	0b001

Accessibility

MRS <Xt>, S3_0_C15_C8_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR6_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUACTLR6_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUACTLR6_EL1;
```

MSR S3_0_C15_C8_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR6_EL1 = X[t];
elsif PSTATE.EL == EL2 then
```

```
    if EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR6_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR6_EL1 = X[t];
```

B.1.22 IMP_CPUACTLR7_EL1, CPU Auxiliary Control Register 7 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-22: AArch64_imp_cpuactlr7_el1 bit assignments

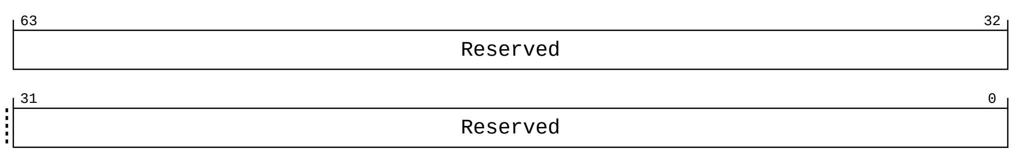


Table B-67: IMP_CPUACTLR7_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_0_C15_C8_2

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_2	0b11	0b000	0b1111	0b1000	0b010

MSR S3_0_C15_C8_2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_2	0b11	0b000	0b1111	0b1000	0b010

Accessibility

MRS <Xt>, S3_0_C15_C8_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR7_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUACTLR7_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUACTLR7_EL1;
```

MSR S3_0_C15_C8_2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR7_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR7_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR7_EL1 = X[t];
```

B.1.23 IMP_ATCR_EL2, CPU Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

0x0

Bit descriptions

Figure B-23: AArch64_imp_atcr_el2 bit assignments

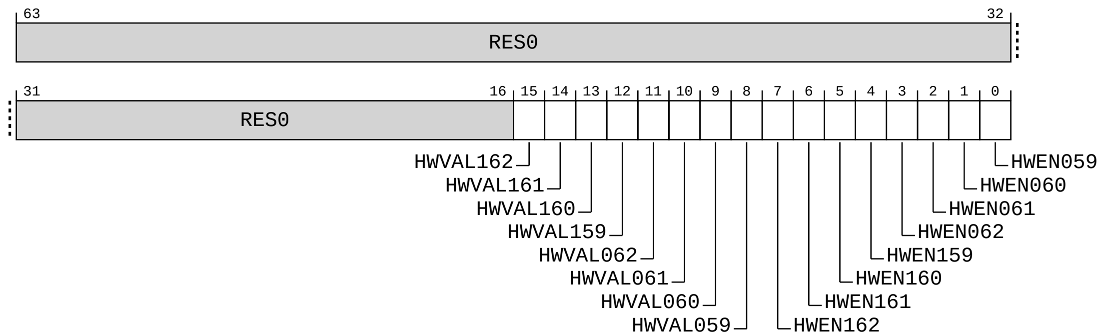


Table B-70: IMP_ATCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to page table walks using TTBR1_EL2 if HWEN162 is set.	0x0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to page table walks using TTBR1_EL2 if HWEN161 is set.	0x0
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to page table walks using TTBR1_EL2 if HWEN160 is set.	0x0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to page table walks using TTBR1_EL2 if HWEN159 is set.	0x0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using TTBR0_EL2 if HWEN062 is set.	0x0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using TTBR0_EL2 if HWEN061 is set.	0x0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using TTBR0_EL2 if HWEN060 is set.	0x0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using TTBR0_EL2 if HWEN059 is set.	0x0
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	0x0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	0x0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	0x0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	0x0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	0x0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	0x0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	0x0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	0x0

Access

MRS <Xt>, S3_4_C15_C7_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_4_C15_C7_0	0b11	0b100	0b1111	0b0111	0b000

MSR S3_4_C15_C7_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_4_C15_C7_0	0b11	0b100	0b1111	0b0111	0b000

Accessibility

MRS <Xt>, S3_4_C15_C7_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return IMP_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.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_ATCR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL2 = X[t];
```

B.1.24 IMP_AVTCR_EL2, CPU Virtualization Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

0x0

Bit descriptions

Figure B-24: AArch64_imp_avtcr_el2 bit assignments

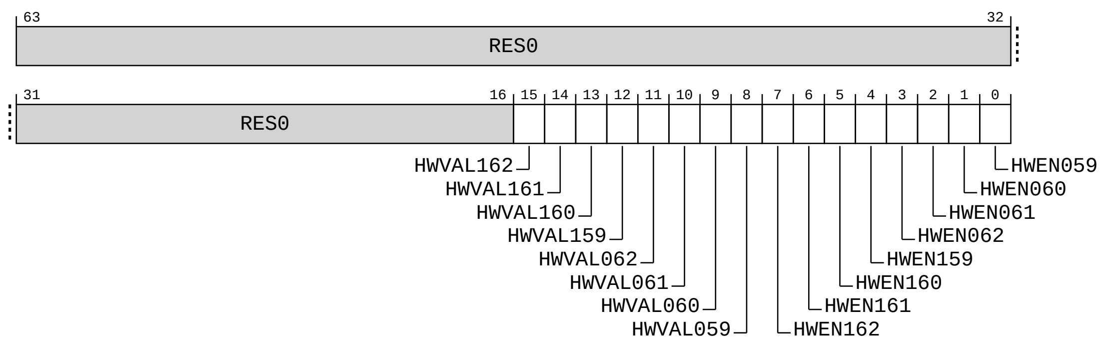


Table B-73: IMP_AVTCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN162 is set.	0x0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN161 is set.	0x0
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN160 is set.	0x0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN159 is set.	0x0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using TTBR0_EL2 if HWEN062 is set.	0x0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using TTBR0_EL2 if HWEN061 is set.	0x0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using TTBR0_EL2 if HWEN060 is set.	0x0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using TTBR0_EL2 if HWEN059 is set.	0x0
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	0x0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	0x0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	0x0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	0x0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	0x0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	0x0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	0x0

Bits	Name	Description	Reset
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	0x0

Access

MRS <Xt>, S3_4_C15_C7_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_4_C15_C7_1	0b11	0b100	0b1111	0b0111	0b001

MSR S3_4_C15_C7_1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_4_C15_C7_1	0b11	0b100	0b1111	0b0111	0b001

Accessibility

MRS <Xt>, S3_4_C15_C7_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return IMP_AVTCR_EL2;
elsif PSTATE.EL == EL3 then
    return IMP_AVTCR_EL2;
```

MSR S3_4_C15_C7_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_AVTCR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_AVTCR_EL2 = X[t];
```

B.1.25 IMP_CPUPPMCR_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-25: AArch64_imp_cpuppmcr_el3 bit assignments

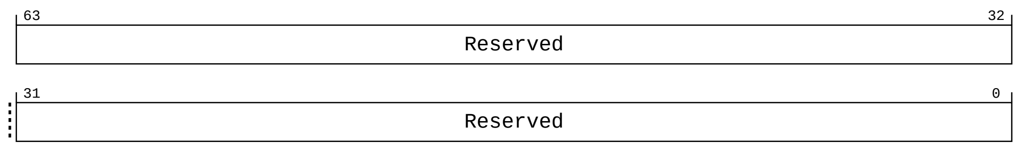


Table B-76: IMP_CPUPPMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C2_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_0	0b11	0b110	0b1111	0b0010	0b000

MSR S3_6_C15_C2_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_0	0b11	0b110	0b1111	0b0010	0b000

Accessibility

MRS <Xt>, S3_6_C15_C2_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR_EL3;
```

MSR S3_6_C15_C2_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR_EL3 = X[t];
```

B.1.26 IMP_CPUPPMCR2_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-26: AArch64_imp_cpuppmcr2_el3 bit assignments

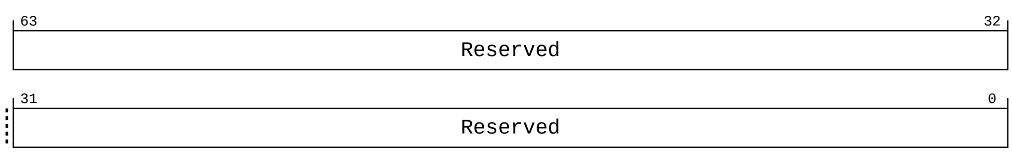


Table B-79: IMP_CPUPPMCR2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C2_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_1	0b11	0b110	0b1111	0b0010	0b001

MSR S3_6_C15_C2_1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_1	0b11	0b110	0b1111	0b0010	0b001

Accessibility

MRS <Xt>, S3_6_C15_C2_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR2_EL3;
```

MSR S3_6_C15_C2_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR2_EL3 = X[t];
```

B.1.27 IMP_CPUPPMCR4_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-27: AArch64_imp_cpuppmcr4_el3 bit assignments

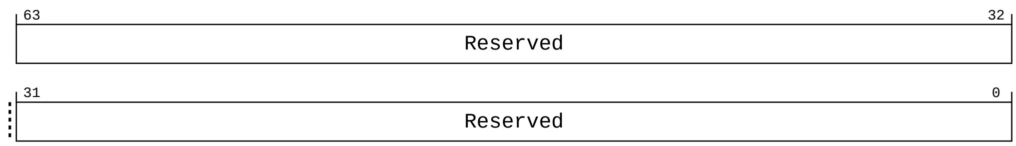


Table B-82: IMP_CPUPPMCR4_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C2_4

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_4	0b11	0b110	0b1111	0b0010	0b100

MSR S3_6_C15_C2_4, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_4	0b11	0b110	0b1111	0b0010	0b100

Accessibility

MRS <Xt>, S3_6_C15_C2_4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR4_EL3;
```

MSR S3_6_C15_C2_4, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR4_EL3 = X[t];
```

B.1.28 IMP_CPUPPMCR5_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-28: AArch64_imp_cpuppmcr5_el3 bit assignments

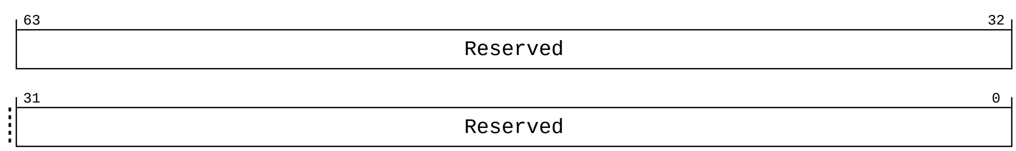


Table B-85: IMP_CPUPPMCR5_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C2_5

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_5	0b11	0b110	0b1111	0b0010	0b101

MSR S3_6_C15_C2_5, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_5	0b11	0b110	0b1111	0b0010	0b101

Accessibility

MRS <Xt>, S3_6_C15_C2_5

```
if PSTATE.EL == EL0 then
```

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR5_EL3;
```

MSR S3_6_C15_C2_5, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR5_EL3 = X[t];
```

B.1.29 IMP_CPUPPMCR6_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-29: AArch64_imp_cpuppmcr6_el3 bit assignments

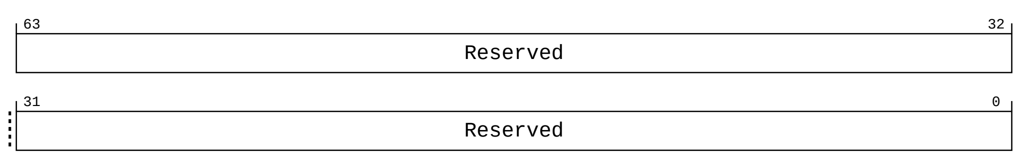


Table B-88: IMP_CPUPPMCR6_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C2_6

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_6	0b11	0b110	0b1111	0b0010	0b110

MSR S3_6_C15_C2_6, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_6	0b11	0b110	0b1111	0b0010	0b110

Accessibility

MRS <Xt>, S3_6_C15_C2_6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR6_EL3;
```

MSR S3_6_C15_C2_6, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR6_EL3 = X[t];
```

B.1.30 IMP_CPUACTLR_EL3, CPU Auxiliary Control Register (EL3)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-30: AArch64_imp_cpuactlr_el3 bit assignments

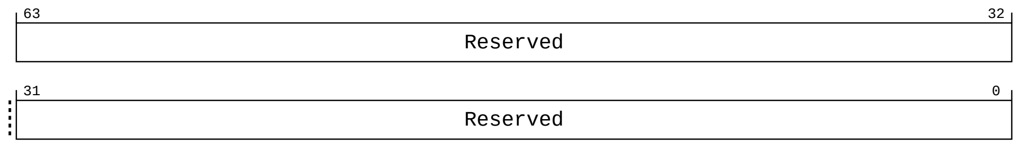


Table B-91: IMP_CPUACTLR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C4_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C4_0	0b11	0b110	0b1111	0b0100	0b000

MSR S3_6_C15_C4_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C4_0	0b11	0b110	0b1111	0b0100	0b000

Accessibility

MRS <Xt>, S3_6_C15_C4_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUACTLR_EL3;
```

MSR S3_6_C15_C4_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR_EL3 = X[t];
```

B.1.31 IMP_ATCR_EL3, CPU Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

`0x0`

Bit descriptions

Figure B-31: AArch64_imp_atcr_el3 bit assignments

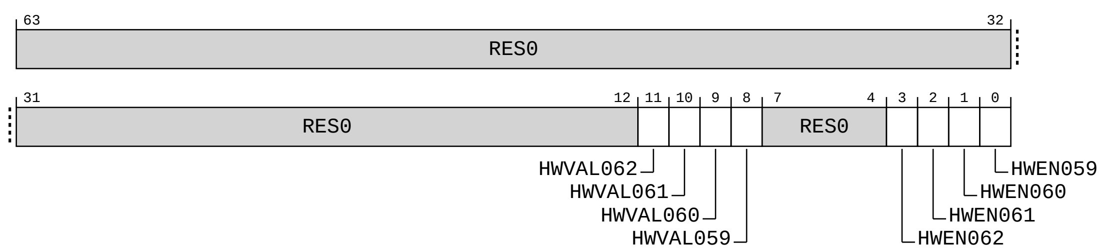


Table B-94: IMP_ATCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	`0x0`
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using TTBR0_EL3 if HWEN062 is set.	`0x0`
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using TTBR0_EL3 if HWEN061 is set.	`0x0`
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using TTBR0_EL3 if HWEN060 is set.	`0x0`
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using TTBR0_EL3 if HWEN059 is set.	`0x0`
[7:4]	RES0	Reserved	`0b0000`
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[3] will be 0 on page table walks.	`0b0`
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[2] will be 0 on page table walks.	`0b0`
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[1] will be 0 on page table walks.	`0b0`
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[0] will be 0 on page table walks.	`0b0`

Access

MRS <Xt>, S3_6_C15_C7_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C7_0	0b11	0b110	0b1111	0b0111	0b000

MSR S3_6_C15_C7_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C7_0	0b11	0b110	0b1111	0b0111	0b000

Accessibility

MRS <Xt>, S3_6_C15_C7_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL3 = X[t];
```

B.1.32 IMP_CPUPSELR_EL3, Selected Instruction Private Select Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-32: AArch64_imp_cpupselr_el3 bit assignments

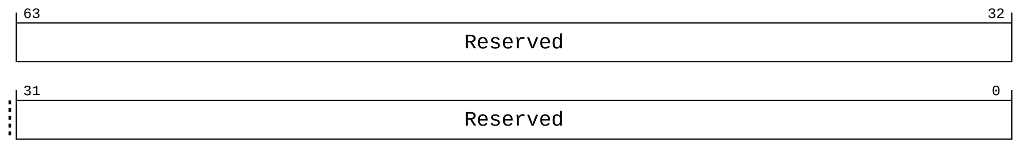


Table B-97: IMP_CPUPSELR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C8_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_0	0b11	0b110	0b1111	0b1000	0b000

MSR S3_6_C15_C8_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_0	0b11	0b110	0b1111	0b1000	0b000

Accessibility

MRS <Xt>, S3_6_C15_C8_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPSELR_EL3 = X[t];
```

B.1.33 IMP_CPUPCR_EL3, Selected Instruction Private Control Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-33: AArch64_imp_cpupcr_el3 bit assignments

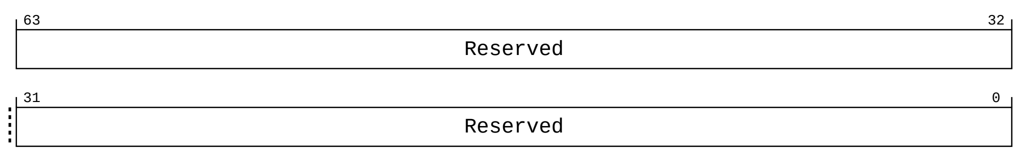


Table B-100: IMP_CPUPCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C8_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_1	0b11	0b110	0b1111	0b1000	0b001

MSR S3_6_C15_C8_1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_1	0b11	0b110	0b1111	0b1000	0b001

Accessibility

MRS <Xt>, S3_6_C15_C8_1

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

```
elsif PSTATE.EL == EL1 then
    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
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPCR_EL3 = X[t];
```

B.1.34 IMP_CPUPOR_EL3, Selected Instruction Private Opcode Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-34: AArch64_imp_cpupor_el3 bit assignments

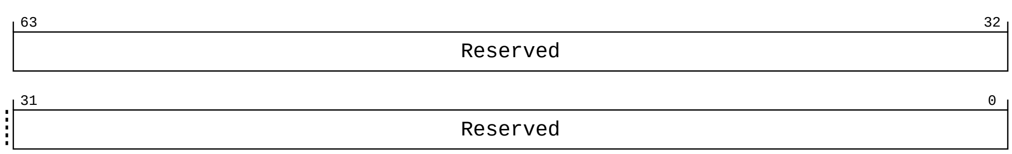


Table B-103: IMP_CPUPOR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C8_2

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_2	0b11	0b110	0b1111	0b1000	0b010

MSR S3_6_C15_C8_2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_2	0b11	0b110	0b1111	0b1000	0b010

Accessibility

MRS <Xt>, S3_6_C15_C8_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPOR_EL3 = X[t];
```

B.1.35 IMP_CPUPMR_EL3, Selected Instruction Private Mask Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-35: AArch64_imp_cpupmr_el3 bit assignments

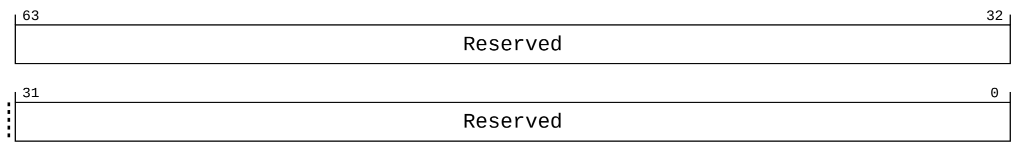


Table B-106: IMP_CPUPMR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C8_3

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_3	0b11	0b110	0b1111	0b1000	0b011

MSR S3_6_C15_C8_3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_3	0b11	0b110	0b1111	0b1000	0b011

Accessibility

MRS <Xt>, S3_6_C15_C8_3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPMR_EL3 = X[t];
```

B.1.36 IMP_CPUPOR2_EL3, Selected Instruction Private Opcode Register 2

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-36: AArch64_imp_cpupor2_el3 bit assignments

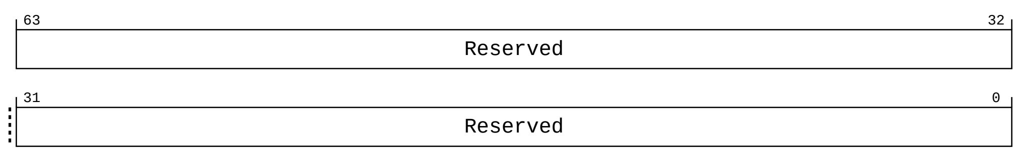


Table B-109: IMP_CPUPOR2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C8_4

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_4	0b11	0b110	0b1111	0b1000	0b100

MSR S3_6_C15_C8_4, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_4	0b11	0b110	0b1111	0b1000	0b100

Accessibility

MRS <Xt>, S3_6_C15_C8_4

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

```
elsif PSTATE.EL == EL1 then
    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
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPOR2_EL3 = X[t];
```

B.1.37 IMP_CPUPMR2_EL3, Selected Instruction Private Mask Register 2

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-37: AArch64_imp_cpupmr2_el3 bit assignments

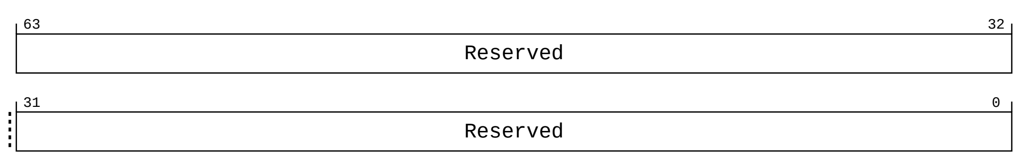


Table B-112: IMP_CPUPMR2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C8_5

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_5	0b11	0b110	0b1111	0b1000	0b101

MSR S3_6_C15_C8_5, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_5	0b11	0b110	0b1111	0b1000	0b101

Accessibility

MRS <Xt>, S3_6_C15_C8_5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPMR2_EL3 = X[t];
```

B.1.38 IMP_CPUPFR_EL3, Selected Instruction Private Flag Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-38: AArch64_imp_cpupfr_el3 bit assignments

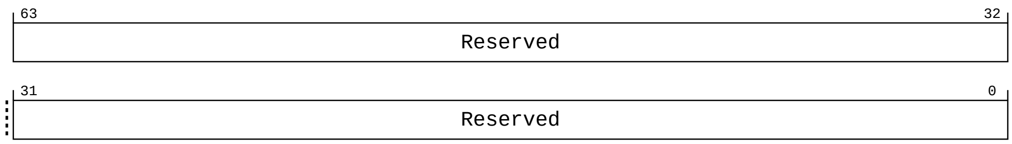


Table B-115: IMP_CPUPFR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3_6_C15_C8_6

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_6	0b11	0b110	0b1111	0b1000	0b110

MSR S3_6_C15_C8_6, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_6	0b11	0b110	0b1111	0b1000	0b110

Accessibility

MRS <Xt>, S3_6_C15_C8_6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPFR_EL3 = X[t];
```

B.1.39 FPCR, Floating-point Control Register

Controls floating-point behavior.

Configurations

The named fields in this register map to the equivalent fields in the AArch32 AArch32-FPSCR.

It is IMPLEMENTATION DEFINED whether the Len and Stride fields can be programmed to non-zero values, which will cause some AArch32 floating-point instruction encodings to be UNDEFINED, or whether these fields are RAZ.

Attributes

Width

64

Functional group

generic-system-control

Reset value

See individual bit resets.

Bit descriptions

Figure B-39: AArch64_fpcr bit assignments

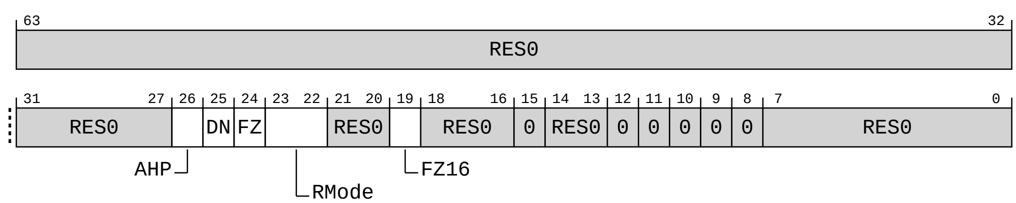


Table B-118: FPCR bit descriptions

Bits	Name	Description	Reset
[63:27]	RES0	Reserved	`0x0`
[26]	AHP	Alternative half-precision control bit: **`0b0`** IEEE half-precision format selected. **`0b1`** Alternative half-precision format selected. This bit is only used for conversions between half-precision floating-point and other floating-point formats. The data-processing instructions added as part of the ARMv8.2-FP16 extension always use the IEEE half-precision format, and ignore the value of this bit.	

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>Reset</th></tr>
<tr><td>[25]</td><td>DN</td><td>Default NaN mode control bit:
<code>0b0</code>
NaN operands propagate through to the output of a floating-point operation.
<code>0b1</code>
Any operation involving one or more NaNs returns the Default NaN.
The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.</td><td></td></tr>
<tr><td>[24]</td><td>FZ</td><td>Flush-to-zero mode control bit.
<code>0b0</code>
Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.
<code>0b1</code>
Flush-to-zero mode enabled.
The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.
This bit has no effect on half-precision calculations.</td><td></td></tr>
<tr><td>[23:22]</td><td>RMode</td><td>Rounding Mode control field.
<code>0b00</code>
Round to Nearest (RN) mode.
<code>0b01</code>
Round towards Plus Infinity (RP) mode.
<code>0b10</code>
Round towards Minus Infinity (RM) mode.
<code>0b11</code>
Round towards Zero (RZ) mode.
The specified rounding mode is used by both scalar and Advanced SIMD floating-point instructions.</td><td></td></tr>
<tr><td>[21:20]</td><td>RES0</td><td>Reserved</td><td><code>0b00</code></td></tr>
<tr><td>[19]</td><td>FZ16</td><td>Flush-to-zero mode control bit on half-precision data-processing instructions.
<code>0b0</code>
Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.
<code>0b1</code>
Flush-to-zero mode enabled.
The value of this bit applies to both scalar and Advanced SIMD floating-point half-precision calculations. A half-precision floating-point number that is flushed to zero as a result of the value of the FZ16 bit does not generate an Input Denormal exception.</td><td></td></tr>
<tr><td>[18:0]</td><td>RES0</td><td>Reserved</td><td><code>0b00000000</code></td></tr>
</table>

Access

MRS <Xt>, FPCR

<systemreg>	op0	op1	CRn	CRm	op2
FPCR	0b11	0b011	0b0100	0b0100	0b000

MSR FPCR, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
FPCR	0b11	0b011	0b0100	0b0100	0b000

Accessibility

MRS <Xt>, FPCR

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

MSR FPCR, <Xt>

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.FPEN != '11' then
```

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

B.1.40 AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL2.

Configurations

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

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

Attributes

Width

64

Functional group

generic-system-control

Reset value

0x0

Bit descriptions

Figure B-40: AArch64_afsr0_el2 bit assignments

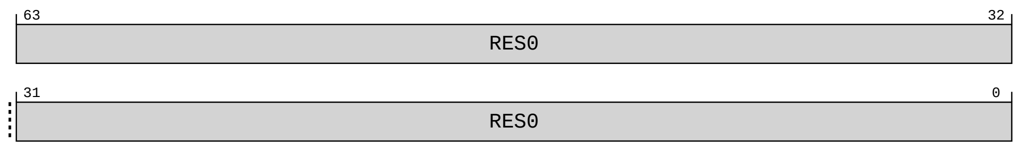


Table B-121: AFSR0_EL2 bit descriptions

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

Access

MRS <Xt>, AFSR0_EL2

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL2	0b11	0b100	0b0101	0b0001	0b000

MSR AFSR0_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL2	0b11	0b100	0b0101	0b0001	0b000

MRS <Xt>, AFSR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL1	0b11	0b000	0b0101	0b0001	0b000

MSR AFSR0_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL1	0b11	0b000	0b0101	0b0001	0b000

Accessibility

MRS <Xt>, AFSR0_EL2

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

MRS <Xt>, AFSR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x128];
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR0_EL2;
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    return AFSR0_EL1;
```

MSR AFSR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x128] = X[t];
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL2 = X[t];
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t];
```

B.1.41 AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL2.

Configurations

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

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

Attributes

Width

64

Functional group

generic-system-control

Reset value

`0x0`

Bit descriptions

Figure B-41: AArch64_afsr1_el2 bit assignments

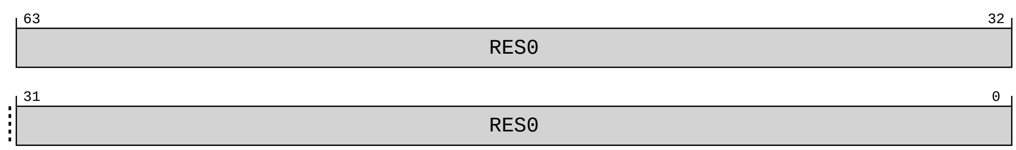


Table B-126: AFSR1_EL2 bit descriptions

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

Access

MRS <Xt>, AFSR1_EL2

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL2	`0b11`	`0b100`	`0b0101`	`0b0001`	`0b001`

MSR AFSR1_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL2	`0b11`	`0b100`	`0b0101`	`0b0001`	`0b001`

MRS <Xt>, AFSR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL1	`0b11`	`0b000`	`0b0101`	`0b0001`	`0b001`

MSR AFSR1_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL1	0b11	0b000	0b0101	0b0001	0b001

Accessibility

MRS <Xt>, AFSR1_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return AFSR1_EL2;
elsif PSTATE.EL == EL3 then
    return AFSR1_EL2;
```

MSR AFSR1_EL2, <Xt>

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

MRS <Xt>, AFSR1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x130];
    else
        return AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR1_EL2;
    else
        return AFSR1_EL1;
elsif PSTATE.EL == EL3 then
    return AFSR1_EL1;
```

MSR AFSR1_EL1, <Xt>

```
if PSTATE.EL == EL0 then
```

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x130] = X[t];
    else
        AFSR1_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL2 = X[t];
    else
        AFSR1_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR1_EL1 = X[t];
```

B.1.42 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

Reset value

`0x0`

Bit descriptions

Figure B-42: AArch64_afsr0_el1 bit assignments

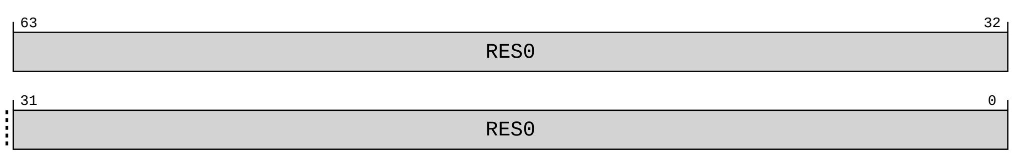


Table B-131: AFSR0_EL1 bit descriptions

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

Access

MRS <Xt>, AFSR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL1	0b11	0b000	0b0101	0b0001	0b000

MSR AFSR0_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL1	0b11	0b000	0b0101	0b0001	0b000

MRS <Xt>, AFSR0_EL12

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL12	0b11	0b101	0b0101	0b0001	0b000

MSR AFSR0_EL12, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL12	0b11	0b101	0b0101	0b0001	0b000

Accessibility

MRS <Xt>, AFSR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x128];
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR0_EL2;
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    return AFSR0_EL1;
```

MSR AFSR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x128] = X[t];
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL2 = X[t];
    else
```

```
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t];
```

MRS <Xt>, AFSR0_EL12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        return NVMem[0x128];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return 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
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x128] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        AFSR0_EL1 = X[t];
    else
        UNDEFINED;
```

B.1.43 AFSR1_EL1, Auxiliary Fault Status Register 1 (EL1)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

`0x0`

Bit descriptions

Figure B-43: AArch64_afsr1_el1 bit assignments

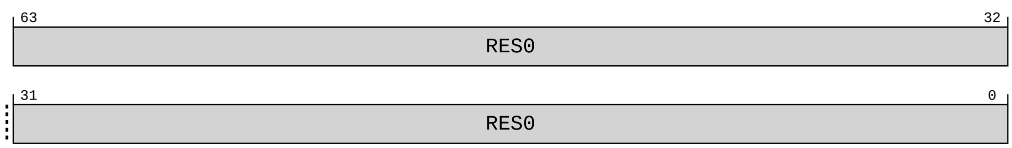


Table B-136: AFSR1_EL1 bit descriptions

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

Access

MRS <Xt>, AFSR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL1	0b11	0b000	0b0101	0b0001	0b001

MSR AFSR1_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL1	0b11	0b000	0b0101	0b0001	0b001

MRS <Xt>, AFSR1_EL12

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL12	0b11	0b101	0b0101	0b0001	0b001

MSR AFSR1_EL12, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL12	0b11	0b101	0b0101	0b0001	0b001

Accessibility

MRS <Xt>, AFSR1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x130];
    else
        return AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR1_EL2;
    else
        return AFSR1_EL1;
elsif PSTATE.EL == EL3 then
    return AFSR1_EL1;
```

MSR AFSR1_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x130] = X[t];
    else
        AFSR1_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL2 = X[t];
    else
        AFSR1_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR1_EL1 = X[t];
```

MRS <Xt>, AFSR1_EL12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        return NVMem[0x130];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR1_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        return AFSR1_EL1;
    else
```

```
        UNDEFINED;
```

MSR AFSR1_EL12, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x130] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        AFSR1_EL1 = X[t];
    else
        UNDEFINED;
```

B.1.44 AFSR0_EL3, Auxiliary Fault Status Register 0 (EL3)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

0x0

Bit descriptions

Figure B-44: AArch64_afsr0_el3 bit assignments

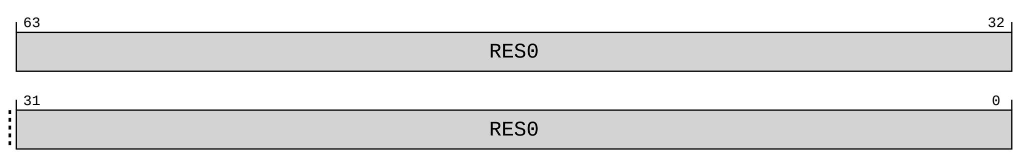


Table B-141: AFSR0_EL3 bit descriptions

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

Access

MRS <Xt>, AFSR0_EL3

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL3	0b11	0b110	0b0101	0b0001	0b000

MSR AFSR0_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL3	0b11	0b110	0b0101	0b0001	0b000

Accessibility

MRS <Xt>, AFSR0_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
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];
```

B.1.45 AFSR1_EL3, Auxiliary Fault Status Register 1 (EL3)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

generic-system-control

Reset value

`0x0`

Bit descriptions

Figure B-45: AArch64_afsr1_el3 bit assignments

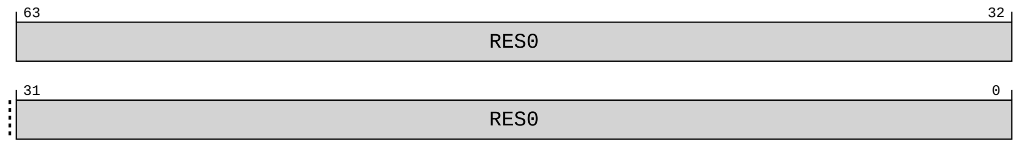


Table B-144: AFSR1_EL3 bit descriptions

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

Access

MRS <Xt>, AFSR1_EL3

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL3	0b11	0b110	0b0101	0b0001	0b001

MSR AFSR1_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL3	0b11	0b110	0b0101	0b0001	0b001

Accessibility

MRS <Xt>, AFSR1_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
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];
```

B.2 AArch64 debug registers

The summary table provides an overview of all implementation defined debug registers in the core. Individual register descriptions provide detailed information.

Table B-147: debug register summary

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

B.2.1 IMP_IDATA0_EL3, Instruction Register 0

Contains data from a preceeding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

debug

Reset value

See individual bit resets.

Bit descriptions

Figure B-46: AArch64_imp_idata0_el3 bit assignments

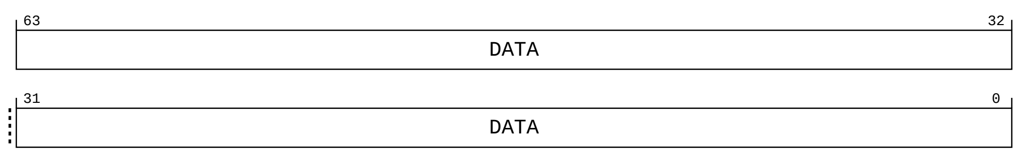


Table B-148: IMP_IDATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

Access

MRS <Xt>, S3_6_C15_C0_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C0_0	0b11	0b110	0b1111	0b0000	0b000

Accessibility

MRS <Xt>, S3_6_C15_C0_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_IDATA0_EL3;
```

B.2.2 IMP_IDATA1_EL3, Instruction Register 0

Contains data from a preceeding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

debug

Reset value

See individual bit resets.

Bit descriptions

Figure B-47: AArch64_imp_idata1_el3 bit assignments

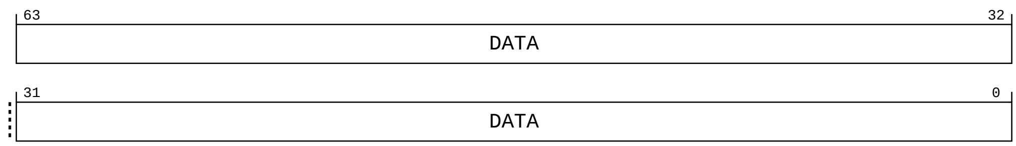


Table B-150: IMP_IDATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

Access

MRS <Xt>, S3_6_C15_C0_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C0_1	0b11	0b110	0b1111	0b0000	0b001

Accessibility

MRS <Xt>, S3_6_C15_C0_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_IDATA1_EL3;
```

B.2.3 IMP_IDATA2_EL3, Instruction Register 0

Contains data from a preceeding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

debug

Reset value

See individual bit resets.

Bit descriptions

Figure B-48: AArch64_imp_idata2_el3 bit assignments

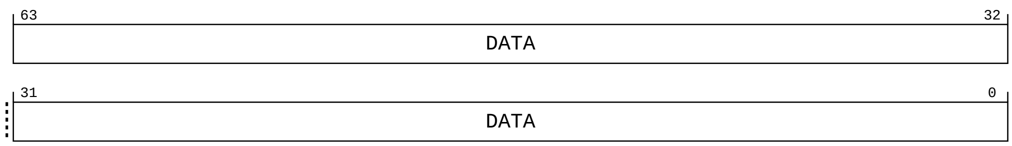


Table B-152: IMP_IDATA2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

Access

MRS <Xt>, S3_6_C15_C0_2

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C0_2	0b11	0b110	0b1111	0b0000	0b010

Accessibility

MRS <Xt>, S3_6_C15_C0_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_IDATA2_EL3;
```

B.2.4 IMP_DDATA0_EL3, Data Register 0

Contains data from a preceeding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

debug

Reset value

See individual bit resets.

Bit descriptions

Figure B-49: AArch64_imp_ddata0_el3 bit assignments

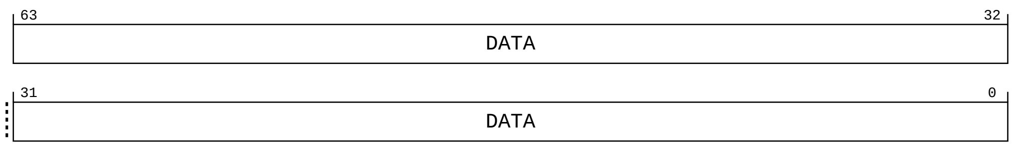


Table B-154: IMP_DDATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

Access

MRS <Xt>, S3_6_C15_C1_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C1_0	0b11	0b110	0b1111	0b0001	0b000

Accessibility

MRS <Xt>, S3_6_C15_C1_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_DDATA0_EL3;
```

B.2.5 IMP_DDATA1_EL3, Data Register 1

Contains data from a preceeding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

debug

Reset value

See individual bit resets.

Bit descriptions

Figure B-50: AArch64_imp_ddata1_el3 bit assignments

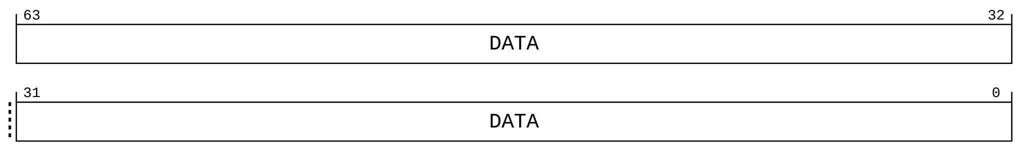


Table B-156: IMP_DDATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

Access

MRS <Xt>, S3_6_C15_C1_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C1_1	0b11	0b110	0b1111	0b0001	0b001

Accessibility

MRS <Xt>, S3_6_C15_C1_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_DDATA1_EL3;
```

B.2.6 IMP_DDATA2_EL3, Data Register 2

Contains data from a preceeding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

debug

Reset value

See individual bit resets.

Bit descriptions

Figure B-51: AArch64_imp_ddata2_el3 bit assignments

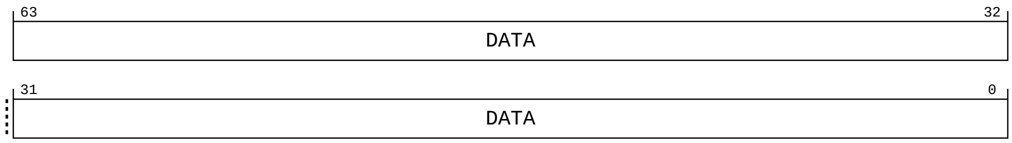


Table B-158: IMP_DDATA2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

Access

MRS <Xt>, S3_6_C15_C1_2

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C1_2	0b11	0b110	0b1111	0b0001	0b010

Accessibility

MRS <Xt>, S3_6_C15_C1_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_DDATA2_EL3;
```

B.3 AArch64 random number control registers

The summary table provides an overview of all implementation defined random number control registers in the core. Individual register descriptions provide detailed information.

Table B-160: random number control register summary

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

B.3.1 IMP_CPURNDBR_EL3, CPU Random Number Base Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

random-number-control

Reset value

`0x0`

Bit descriptions

Figure B-52: AArch64_imp_cpurndbr_el3 bit assignments

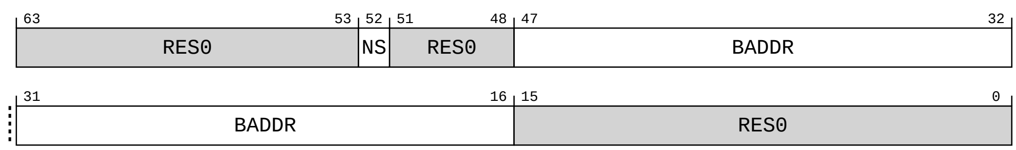


Table B-161: IMP_CPURNDBR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:53]	RES0	Reserved	`0x0`
[52]	NS	Indicates the security state of the external RNG block accesses. The possible values are: **`0b0`** Secure **`0b1`** Non-secure	`0x0`
[51:48]	RES0	Reserved	`0b0000`
[47:16]	BADDR	Indicates the base address bits [47:16] of the external RNG block	`0x0`
[15:0]	RES0	Reserved	`0x0`

Access

MRS <Xt>, S3_6_C15_C3_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C3_0	`0b11`	`0b110`	`0b1111`	`0b0011`	`0b000`

MSR S3_6_C15_C3_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C3_0	0b11	0b110	0b1111	0b0011	0b000

Accessibility

MRS <Xt>, S3_6_C15_C3_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPURNDBR_EL3;
```

MSR S3_6_C15_C3_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPURNDBR_EL3 = X[t];
```

B.3.2 IMP_CPURNDPEID_EL3, CPU Random Number Packet Identification Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

random-number-control

Reset value

0x0

Bit descriptions

Figure B-53: AArch64_imp_cpurndpeid_el3 bit assignments

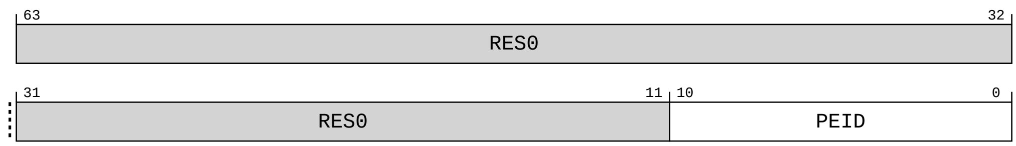


Table B-164: IMP_CPURNDPEID_EL3 bit descriptions

Bits	Name	Description	Reset
[63:11]	RES0	Reserved	0x0
[10:0]	PEID	Unique 11-bit hardware identification which is used to construct the address for RNDR accesses: RNDR address={CPURNDBR_EL3[47:16],CPURNDPEID_EL3[10:0],1'b0,4'b0}, RNDRRS address={CPURNDBR_EL3[47:16],CPURNDPEID_EL3[10:0],1'b1,4'b0}	0x0

Access

MRS <Xt>, S3_6_C15_C3_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C3_1	0b11	0b110	0b1111	0b0011	0b001

MSR S3_6_C15_C3_1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C3_1	0b11	0b110	0b1111	0b0011	0b001

Accessibility

MRS <Xt>, S3_6_C15_C3_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPURNDPEID_EL3;
```

MSR S3_6_C15_C3_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPURNDPEID_EL3 = X[t];
```

B.4 AArch64 system instruction register summary

The summary table provides an overview of all implementation defined system instruction registers in the core. Individual register descriptions provide detailed information.

Table B-167: system instruction register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
SYS_IMP_RAMINDEX	1	C15	6	C0	0	See individual bit resets.	64-bit	RAM Index

B.4.1 SYS_IMP_RAMINDEX, RAM Index

Read contents of the cache specified by the source register into AArch64-IMP_IDATA0_EL3, AArch64-IMP_IDATA1_EL3, AArch64-IMP_IDATA2_EL3, AArch64-IMP_DDATA0_EL3, AArch64-IMP_DDATA1_EL3, and AArch64-IMP_DDATA2_EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

system-instruction

Reset value

See individual bit resets.

Bit descriptions

Figure B-54: AArch64_sys_imp_ramindex bit assignments

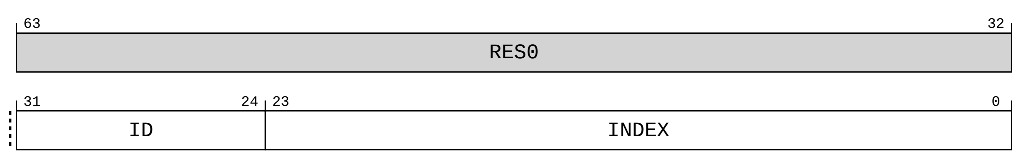


Table B-168: SYS_IMP_RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:24]	ID	RAM ID (See Chapter 10)	
[23:0]	INDEX	RAM Index (See Chapter 10)	

Access

Accesses to this instruction use the following encodings:

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

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C0_0	0b01	0b110	0b1111	0b0000	0b000

Accessibility

Accesses to this instruction use the following encodings:

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

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

B.5 AArch64 identification registers

The summary table provides an overview of all implementation defined identification registers in the core. Individual register descriptions provide detailed information.

Table B-170: identification register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
MIDR_EL1	3	C0	0	C0	0	See individual bit resets.	64-bit	Main ID Register
MPIDR_EL1	3	C0	0	C0	5	See individual bit resets.	64-bit	Multiprocessor Affinity Register
REVIDR_EL1	3	C0	0	C0	6	See individual bit resets.	64-bit	Revision ID Register
ID_PFR0_EL1	3	C0	0	C1	0	See individual bit resets.	64-bit	AArch32 Processor Feature Register 0
ID_PFR1_EL1	3	C0	0	C1	1	See individual bit resets.	64-bit	AArch32 Processor Feature Register 1
ID_DFR0_EL1	3	C0	0	C1	2	See individual bit resets.	64-bit	AArch32 Debug Feature Register 0
ID_AFR0_EL1	3	C0	0	C1	3	0x0	64-bit	AArch32 Auxiliary Feature Register 0
ID_MMFR0_EL1	3	C0	0	C1	4	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 0
ID_MMFR1_EL1	3	C0	0	C1	5	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 1
ID_MMFR2_EL1	3	C0	0	C1	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 2
ID_MMFR3_EL1	3	C0	0	C1	7	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 3
ID_ISAR0_EL1	3	C0	0	C2	0	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 0
ID_ISAR1_EL1	3	C0	0	C2	1	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 1
ID_ISAR2_EL1	3	C0	0	C2	2	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 2
ID_ISAR3_EL1	3	C0	0	C2	3	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 3
ID_ISAR4_EL1	3	C0	0	C2	4	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 4
ID_ISAR5_EL1	3	C0	0	C2	5	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 5
ID_MMFR4_EL1	3	C0	0	C2	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 4

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
ID_ISAR6_EL1	3	C0	0	C2	7	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 6
MVFR0_EL1	3	C0	0	C3	0	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 0
MVFR1_EL1	3	C0	0	C3	1	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 1
MVFR2_EL1	3	C0	0	C3	2	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 2
ID_PFR2_EL1	3	C0	0	C3	4	See individual bit resets.	64-bit	AArch32 Processor Feature Register 2
ID_DFR1_EL1	3	C0	0	C3	5	`0x0`	64-bit	Debug Feature Register 1
ID_AA64PFR0_EL1	3	C0	0	C4	0	See individual bit resets.	64-bit	AArch64 Processor Feature Register 0
ID_AA64PFR1_EL1	3	C0	0	C4	1	See individual bit resets.	64-bit	AArch64 Processor Feature Register 1
ID_AA64ZFR0_EL1	3	C0	0	C4	4	See individual bit resets.	64-bit	SVE Feature ID register 0
ID_AA64DFR0_EL1	3	C0	0	C5	0	See individual bit resets.	64-bit	AArch64 Debug Feature Register 0
ID_AA64DFR1_EL1	3	C0	0	C5	1	`0x0`	64-bit	AArch64 Debug Feature Register 1
ID_AA64AFR0_EL1	3	C0	0	C5	4	`0x0`	64-bit	AArch64 Auxiliary Feature Register 0
ID_AA64AFR1_EL1	3	C0	0	C5	5	`0x0`	64-bit	AArch64 Auxiliary Feature Register 1
ID_AA64ISAR0_EL1	3	C0	0	C6	0	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 0
ID_AA64ISAR1_EL1	3	C0	0	C6	1	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 1
ID_AA64MMFR0_EL1	3	C0	0	C7	0	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 0
ID_AA64MMFR1_EL1	3	C0	0	C7	1	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 1
ID_AA64MMFR2_EL1	3	C0	0	C7	2	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 2
CLIDR_EL1	3	C0	1	C0	1	See individual bit resets.	64-bit	Cache Level ID Register
GMID_EL1	3	C0	1	C0	4	See individual bit resets.	64-bit	Multiple tag transfer ID register
CTR_EL0	3	C0	3	C0	1	See individual bit resets.	64-bit	Cache Type Register
DCZID_EL0	3	C0	3	C0	7	See individual bit resets.	64-bit	Data Cache Zero ID register
MPAMIDR_EL1	3	C10	0	C4	4	See individual bit resets.	64-bit	MPAM ID Register (EL1)
IMP_CPUCFR_EL1	3	C15	0	C0	0	See individual bit resets.	64-bit	CPU Configuration Register

B.5.1 MIDR_EL1, Main ID Register

Provides identification information for the PE, including an implementer code for the device and a device ID number.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-55: AArch64_midr_el1 bit assignments

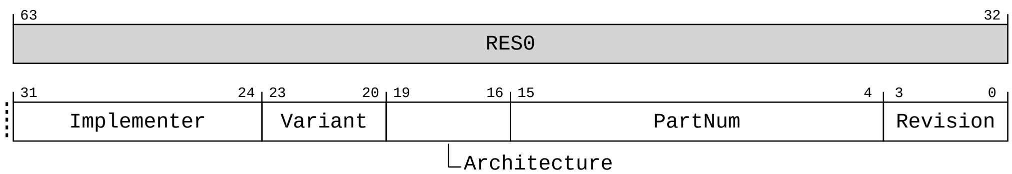


Table B-171: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:24]	Implementer	Indicates the implementer code. This value is: **`0b01000001`** Arm Limited	
[23:20]	Variant	An **IMPLEMENTATION DEFINED** variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product. **`0b0000`** r0p0	
[19:16]	Architecture	Indicates the architecture code. This value is: **`0b1111`** Architecture is defined by ID registers	
[15:4]	PartNum	An **IMPLEMENTATION DEFINED** primary part number for the device. On processors implemented by Arm, if the top four bits of the primary part number are `0x0` or `0x7`, the variant and architecture are encoded differently. **`0b110101001001`** Neoverse N2	
[3:0]	Revision	An **IMPLEMENTATION DEFINED** revision number for the device. **`0b0000`** r0p0	

Access

MRS <Xt>, MIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MIDR_EL1	`0b11`	`0b000`	`0b0000`	`0b0000`	`0b000`

Accessibility

MRS <Xt>, MIDR_EL1

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

```
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.MIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() then
        return VPIDR_EL2;
    else
        return MIDR_EL1;
elsif PSTATE.EL == EL2 then
    return MIDR_EL1;
elsif PSTATE.EL == EL3 then
    return MIDR_EL1;
```

B.5.2 MPIDR_EL1, Multiprocessor Affinity Register

In a multiprocessor system, provides an additional PE identification mechanism for scheduling purposes.

Configurations

In a uniprocessor system Arm recommends that each Aff<n> field of this register returns a value of 0.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-56: AArch64_mpidr_el1 bit assignments

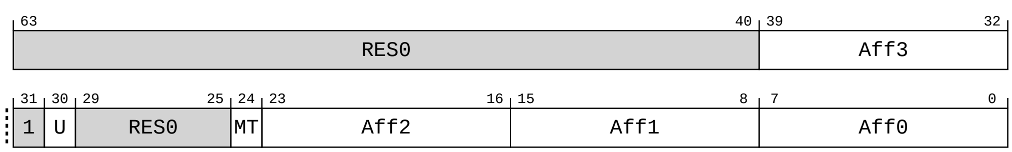


Table B-173: MPIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	0x0
[39:32]	Aff3	Affinity level 3. See the description of Aff0 for more information. Aff3 is not supported in AArch32 state. The value will be determined by the CLUSTERIDAFF3 configuration pins.	
[31]	RES1	Reserved	0b1

Bits	Name	Description	Reset
[30]	U	Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system. The possible values of this bit are: **0b0** Processor is part of a multiprocessor system.	
[29:25]	RES0	Reserved	0b00000
[24]	MT	Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. See the description of Aff0 for more information about affinity levels. The possible values of this bit are: **0b1** Performance of PEs at the lowest affinity level, or PEs with MPIDR_EL1.MT set to 1, different affinity level 0 values, and the same values for affinity level 1 and higher, is very interdependent.	
[23:16]	Aff2	Affinity level 2. See the description of Aff0 for more information. The value will be determined by the CLUSTERIDAFF2 configuration pins.	
[15:8]	Aff1	Affinity level 1. See the description of Aff0 for more information. Value read from the CPUID configuration pins. Identification number for each CPU in an cluster counting from zero.	
[7:0]	Aff0	Affinity level 0. This is the affinity level that is most significant for determining PE behavior. Higher affinity levels are increasingly less significant in determining PE behavior. The assigned value of the MPIDR.{Aff2, Aff1, Aff0} or AArch64-MPIDR_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole. **0b00000000** Only one thread.	

Access

MRS <Xt>, MPIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MPIDR_EL1	0b11	0b000	0b0000	0b0000	0b101

Accessibility

MRS <Xt>, MPIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.MPIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() then
        return VMPIDR_EL2;
    else
        return MPIDR_EL1;
elsif PSTATE.EL == EL2 then
    return MPIDR_EL1;
elsif PSTATE.EL == EL3 then
    return MPIDR_EL1;
```

B.5.3 REVIDR_EL1, Revision ID Register

The REVIDR_EL1 provides revision information, additional to MIDR_EL1, that identifies minor fixes (errata) which might be present in a specific implementation of the Neoverse N2 core. Refer to the Neoverse N2 Product Errata Notice (PEN) for information on how to interpret the values in this register.

Configurations

If REVIDR_EL1 has the same value as AArch64-MIDR_EL1, then its contents have no significance.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-57: AArch64_revidr_el1 bit assignments

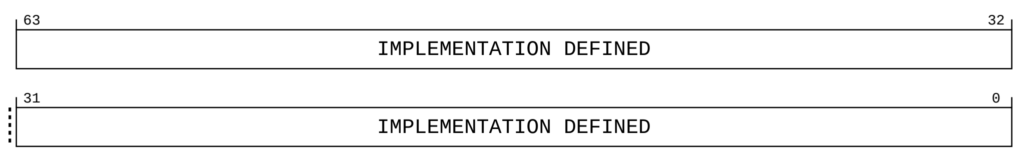


Table B-175: REVIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED	

Access

MRS <Xt>, REVIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
REVIDR_EL1	0b11	0b000	0b0000	0b0000	0b110

Accessibility

MRS <Xt>, REVIDR_EL1

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

```
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.REVIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return REVIDR_EL1;
elsif PSTATE.EL == EL2 then
    return REVIDR_EL1;
elsif PSTATE.EL == EL3 then
    return REVIDR_EL1;
```

B.5.4 ID_PFR0_EL1, AArch32 Processor Feature Register 0

Gives top-level information about the instruction sets supported by the PE in AArch32 state.

Must be interpreted with AArch64-ID_PFR1_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-58: AArch64_id_pfr0_el1 bit assignments

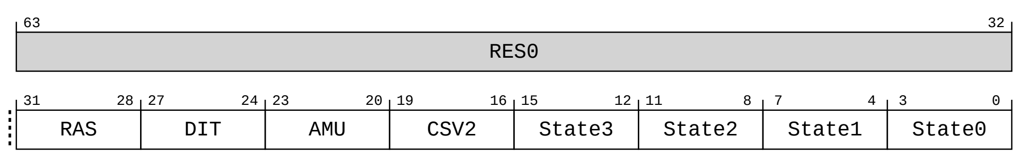


Table B-177: ID_PFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:28]	RAS	RAS Extension version. Defined values are: **0b0010** ARMv8.4-RAS present. As 0b0001, and adds support for additional ERXMISC<m> System registers. Error records accessed through System registers conform to RAS System Architecture v1.1, which includes simplifications to ext-ERR<n>STATUS and support for the optional RAS Timestamp Extension.	

Bits	Name	Description	Reset
[27:24]	DIT	Data Independent Timing. Defined values are: **0b0001** AArch32 provides the CPSR.DIT mechanism to guarantee constant execution time of certain instructions.	
[23:20]	AMU	Activity Monitors Extension. Defined values are: **0b0001** AMUv1 for Armv8.4 is implemented.	
[19:16]	CSV2	Speculative use of out of context branch targets. Defined values are: **0b0001** Branch targets trained in one hardware described context can only affect speculative execution in a different hardware described context in a hard-to-determine way.	
[15:12]	State3	T32EE instruction set support. Defined values are: **0b0000** Not implemented.	
[11:8]	State2	Jazelle extension support. Defined values are: **0b0001** Jazelle extension implemented, without clearing of AArch32-JOSCR.CV on exception entry.	
[7:4]	State1	T32 instruction set support. Defined values are: **0b0011** T32 encodings after the introduction of Thumb-2 technology implemented, for all 16-bit and 32-bit T32 basic instructions.	
[3:0]	State0	A32 instruction set support. Defined values are: **0b0001** A32 instruction set implemented.	

Access

MRS <Xt>, ID_PFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_PFR0_EL1	0b11	0b000	0b0000	0b0001	0b000

Accessibility

MRS <Xt>, ID_PFR0_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_PFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_PFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_PFR0_EL1;
```

B.5.5 ID_PFR1_EL1, AArch32 Processor Feature Register 1

Gives information about the AArch32 programmers' model.

Must be interpreted with AArch64-ID_PFR0_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-59: AArch64_id_pfr1_el1 bit assignments

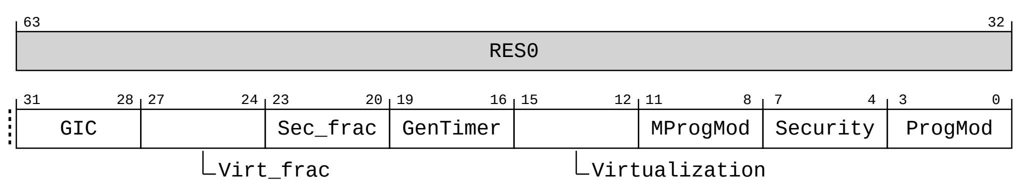


Table B-179: ID_PFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	GIC	System register GIC CPU interface. Defined values are: **`0b0000`** When GICCDISABLE is HIGH, GIC CPU interface is disabled. **`0b0011`** When Port GICCDISABLE is Low, GIC (version 4.1) CPU interface is enabled.	
[27:24]	Virt_frac	Virtualization fractional field. When the Virtualization field is 0000, determines the support for features from the ARMv7 Virtualization Extensions. Defined values are: **`0b0000`** No features from the ARMv7 Virtualization Extensions are implemented.	

Bits	Name	Description	Reset
[23:20]	Sec_frac	Security fractional field. When the Security field is 0000, determines the support for features from the ARMv7 Security Extensions. Defined values are: **0b0000** No features from the ARMv7 Security Extensions are implemented.	
[19:16]	GenTimer	Generic Timer support. Defined values are: **0b0001** Generic Timer is implemented.	
[15:12]	Virtualization	Virtualization support. Defined values are: **0b0000** EL2, Hyp mode, and the HVC instruction not implemented.	
[11:8]	MProgMod	M profile programmers' model support. Defined values are: **0b0000** Not supported.	
[7:4]	Security	Security support. Defined values are: **0b0000** EL3, Monitor mode, and the SMC instruction not implemented.	
[3:0]	ProgMod	Support for the standard programmers' model for Armv4 and later. Model must support User, FIQ, IRQ, Supervisor, Abort, Undefined, and System modes. Defined values are: **0b0000** Not supported.	

Access

MRS <Xt>, ID_PFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_PFR1_EL1	0b11	0b000	0b0000	0b0001	0b001

Accessibility

MRS <Xt>, ID_PFR1_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_PFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_PFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_PFR1_EL1;
```

B.5.6 ID_DFR0_EL1, AArch32 Debug Feature Register 0

Provides top level information about the debug system in AArch32 state.

Must be interpreted with the Main ID Register, AArch64-MIDR_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-60: AArch64_id_dfr0_el1 bit assignments

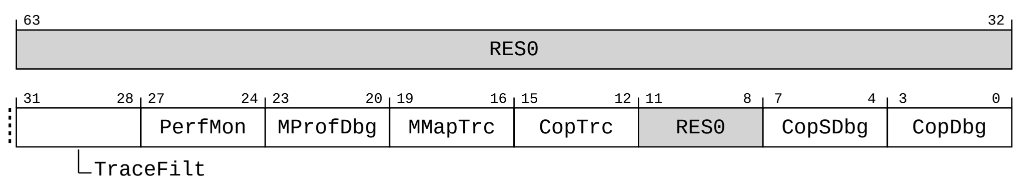


Table B-181: ID_DFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. Defined values are: **`0b0001`** Armv8.4 Self-hosted Trace Extension implemented.	
[27:24]	PerfMon	Performance Monitors Extension version. This field does not follow the standard ID scheme, but uses the Alternative ID scheme described in 'Alternative ID scheme used for the Performance Monitors Extension version'. Defined values are: **`0b0110`** PMUv3 Implemented Armv8.5.	
[23:20]	MProfDbg	M Profile Debug. Support for memory-mapped debug model for M profile processors. Defined values are: **`0b0000`** Not supported.	

Bits	Name	Description	Reset
[19:16]	MMapTrc	Memory Mapped Trace. Support for memory-mapped trace model. Defined values are: **0b0001** Support for Arm trace architecture, with memory-mapped access.	
[15:12]	CopTrc	Support for System registers-based trace model, using registers in the coproc == 1110 encoding space. Defined values are: **0b0001** Support for Arm trace architecture, with System registers access.	
[11:8]	RES0	Reserved	0b0000
[7:4]	CopSDbg	Support for a System registers-based Secure debug model, using registers in the coproc = 1110 encoding space, for an A profile processor that includes EL3. If EL3 is not implemented and the implemented Security state is Non-secure state, this field is RES0. Otherwise, this field reads the same as bits [3:0]. **0b1001** As per CopDbg	
[3:0]	CopDbg	Support for System registers-based debug model, using registers in the coproc == 1110 encoding space, for A and R profile processors. Defined values are: **0b1001** Support for Armv8.4 debug architecture.	

Access

MRS <Xt>, ID_DFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_DFR0_EL1	0b11	0b000	0b0000	0b0001	0b010

Accessibility

MRS <Xt>, ID_DFR0_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_DFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_DFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_DFR0_EL1;
```

B.5.7 ID_AFR0_EL1, AArch32 Auxiliary Feature Register 0

Provides information about the **IMPLEMENTATION DEFINED** features of the PE in AArch32 state.

Must be interpreted with the Main ID Register, AArch64-MIDR_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

`0x0`

Bit descriptions

Figure B-61: AArch64_id_afr0_el1 bit assignments

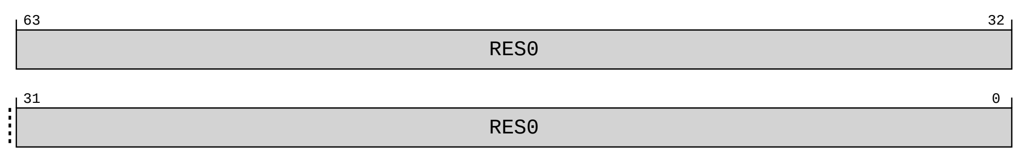


Table B-183: ID_AFR0_EL1 bit descriptions

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

Access

MRS <Xt>, ID_AFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AFR0_EL1	`0b11`	`0b000`	`0b0000`	`0b0001`	`0b011`

Accessibility

MRS <Xt>, ID_AFR0_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_AFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AFR0_EL1;
```

B.5.8 ID_MMFR0_EL1, AArch32 Memory Model Feature Register 0

Provides information about the implemented memory model and memory management support in AArch32 state.

Must be interpreted with AArch64-ID_MMFR1_EL1, AArch64-ID_MMFR2_EL1, AArch64-ID_MMFR3_EL1, and AArch64-ID_MMFR4_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-62: AArch64_id_mmfr0_el1 bit assignments

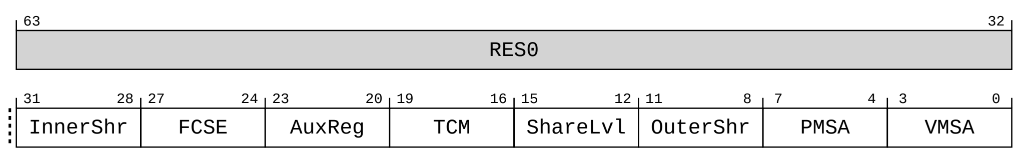


Table B-185: ID_MMFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	InnerShr	Innermost Shareability. Indicates the innermost shareability domain implemented. Defined values are: **`0b0001`** Implemented with hardware coherency support.	

Bits	Name	Description	Reset
[27:24]	FCSE	Indicates whether the implementation includes the FCSE. Defined values are: **0b0000** Not supported.	
[23:20]	AuxReg	Auxiliary Registers. Indicates support for Auxiliary registers. Defined values are: **0b0010** Support for Auxiliary Fault Status Registers (AArch32-AIFSR and AArch32-ADFSR) and Auxiliary Control Register.	
[19:16]	TCM	Indicates support for TCMs and associated DMAs. Defined values are: **0b0000** Not supported.	
[15:12]	ShareLvl	Shareability Levels. Indicates the number of shareability levels implemented. Defined values are: **0b0001** Two levels of shareability implemented.	
[11:8]	OuterShr	Outermost Shareability. Indicates the outermost shareability domain implemented. Defined values are: **0b0001** Implemented with hardware coherency support.	
[7:4]	PMSA	Indicates support for a PMSA. Defined values are: **0b0000** Not supported.	
[3:0]	VMSA	Indicates support for a VMSA. Defined values are: **0b0101** Support for VMSAv7, with support for remapping and the Access flag; The PXN bit in the Short-descriptor translation table format descriptors and the Long-descriptor translation table format	

Access

MRS <Xt>, ID_MMFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_MMFR0_EL1	0b11	0b000	0b0000	0b0001	0b100

Accessibility

MRS <Xt>, ID_MMFR0_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_MMFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_MMFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_MMFR0_EL1;
```

B.5.9 ID_MMFR1_EL1, AArch32 Memory Model Feature Register 1

Provides information about the implemented memory model and memory management support in AArch32 state.

Must be interpreted with AArch64-ID_MMFR0_EL1, AArch64-ID_MMFR2_EL1, AArch64-ID_MMFR3_EL1, and AArch64-ID_MMFR4_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-63: AArch64_id_mmfr1_el1 bit assignments

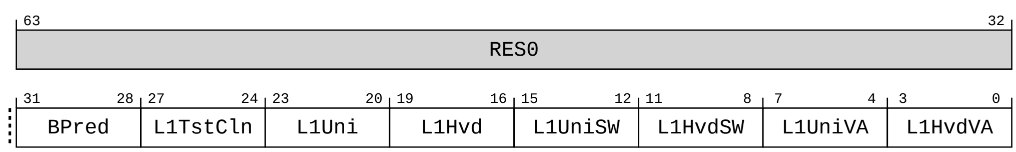


Table B-187: ID_MMFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	BPred	Branch Predictor. Indicates branch predictor management requirements. Defined values are: **`0b0100`** For execution correctness, branch predictor requires no flushing at any time.	
[27:24]	L1TstCln	Level 1 cache Test and Clean. Indicates the supported Level 1 data cache test and clean operations, for Harvard or unified cache implementations. Defined values are: **`0b0000`** None supported.	
[23:20]	L1Uni	Level 1 Unified cache. Indicates the supported entire Level 1 cache maintenance operations for a unified cache implementation. Defined values are: **`0b0000`** None supported.	

Bits	Name	Description	Reset
[19:16]	L1Hvd	Level 1 Harvard cache. Indicates the supported entire Level 1 cache maintenance operations for a Harvard cache implementation. Defined values are: **0b0000** None supported.	
[15:12]	L1UniSW	Level 1 Unified cache by Set/Way. Indicates the supported Level 1 cache line maintenance operations by set/way, for a unified cache implementation. Defined values are: **0b0000** None supported.	
[11:8]	L1HvdSW	Level 1 Harvard cache by Set/Way. Indicates the supported Level 1 cache line maintenance operations by set/way, for a Harvard cache implementation. Defined values are: **0b0000** None supported.	
[7:4]	L1UniVA	Level 1 Unified cache by Virtual Address. Indicates the supported Level 1 cache line maintenance operations by VA, for a unified cache implementation. Defined values are: **0b0000** None supported.	
[3:0]	L1HvdVA	Level 1 Harvard cache by Virtual Address. Indicates the supported Level 1 cache line maintenance operations by VA, for a Harvard cache implementation. Defined values are: **0b0000** None supported.	

Access

MRS <Xt>, ID_MMFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_MMFR1_EL1	0b11	0b000	0b0000	0b0001	0b101

Accessibility

MRS <Xt>, ID_MMFR1_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_MMFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_MMFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_MMFR1_EL1;
```

B.5.10 ID_MMFR2_EL1, AArch32 Memory Model Feature Register 2

Provides information about the implemented memory model and memory management support in AArch32 state.

Must be interpreted with AArch64-ID_MMFR0_EL1, AArch64-ID_MMFR1_EL1, AArch64-ID_MMFR3_EL1, and AArch64-ID_MMFR4_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-64: AArch64_id_mmfr2_el1 bit assignments

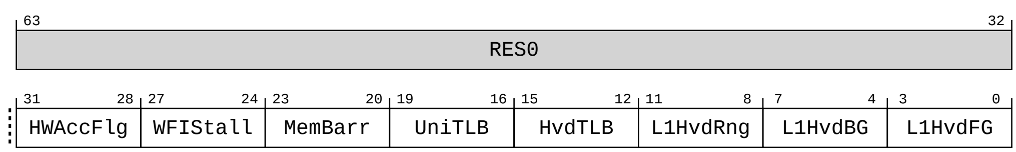


Table B-189: ID_MMFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	HWAccFlg	Hardware Access Flag. In earlier versions of the Arm Architecture, this field indicates support for a Hardware Access flag, as part of the VMSAv7 implementation. Defined values are: **`0b0000`** Not supported.	
[27:24]	WFIStall	Wait For Interrupt Stall. Indicates the support for Wait For Interrupt (WFI) stalling. Defined values are: **`0b0001`** Support for WFI stalling.	
[23:20]	MemBarr	Memory Barrier. Indicates the supported memory barrier System instructions in the (coproc==1111) encoding space: **`0b0010`** Supported memory barrier System instructions are Data Synchronization Barrier (DSB), Instruction Synchronization Barrier (ISB) and Data Memory Barrier (DMB).	

Bits	Name	Description	Reset
[19:16]	UniTLB	Unified TLB. Indicates the supported TLB maintenance operations, for a unified TLB implementation. Defined values are: **0b0110** Supported unified TLB maintenance operations are - Invalidate all entries in the TLB. - Invalidate TLB entry by VA. - Invalidate TLB entries by ASID match. - Invalidate instruction TLB and data TLB entries by VA All ASID. This is a shared unified TLB operation. - Invalidate Hyp mode unified TLB entry by VA. - Invalidate entire Non-secure PL1 and PL0 unified TLB. - Invalidate entire Hyp mode unified TLB. - TLBIMVALIS, TLBIMVAALIS, TLBIMVALHIS, TLBIMVAL, TLBIMVAAL, and TLBIMVALH. - TLBIIPAS2IS, TLBIIPAS2LIS, TLBIIPAS2, and TLBIIPAS2L.	
[15:12]	HvdTLB	Harvard TLB. Indicates the supported TLB maintenance operations, for a Harvard TLB implementation: **0b0000** Not supported.	
[11:8]	L1HvdRng	Level 1 Harvard cache Range. Indicates the supported Level 1 cache maintenance range operations, for a Harvard cache implementation. Defined values are: **0b0000** Not supported.	
[7:4]	L1HvdBG	Level 1 Harvard cache Background fetch. Indicates the supported Level 1 cache background fetch operations, for a Harvard cache implementation. **0b0000** Not supported.	
[3:0]	L1HvdFG	L1 Harvard cache Foreground fetch. Indicates the supported L1 cache foreground prefetch operations, for a Harvard cache implementation **0b0000** Not supported.	

Access

MRS <Xt>, ID_MMFR2_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_MMFR2_EL1	0b11	0b000	0b0000	0b0001	0b110

Accessibility

MRS <Xt>, ID_MMFR2_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_MMFR2_EL1;
elsif PSTATE.EL == EL2 then
    return ID_MMFR2_EL1;
elsif PSTATE.EL == EL3 then
    return ID_MMFR2_EL1;
```

B.5.11 ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3

Provides information about the implemented memory model and memory management support in AArch32 state.

Must be interpreted with AArch64-ID_MMFR0_EL1, AArch64-ID_MMFR1_EL1, AArch64-ID_MMFR2_EL1, and AArch64-ID_MMFR4_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-65: AArch64_id_mmfr3_el1 bit assignments

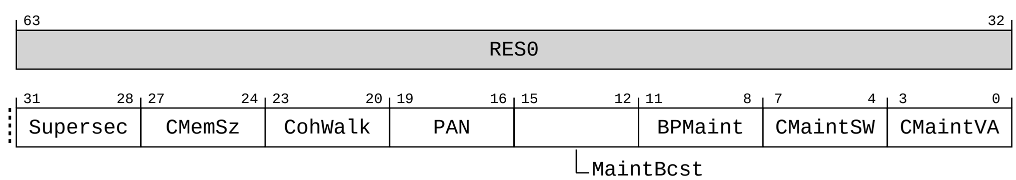


Table B-191: ID_MMFR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	Supersec	Supersections. On a VMSA implementation, indicates whether Supersections are supported. Defined values are: **`0b0000`** Supersections supported.	
[27:24]	CMemSz	Cached Memory Size. Indicates the physical memory size supported by the caches. Defined values are: **`0b0010`** 1TB or more, corresponding to a 40-bit or larger physical address range.	
[23:20]	CohWalk	Coherent Walk. Indicates whether Translation table updates require a clean to the Point of Unification. Defined values are: **`0b0001`** Updates to the translation tables do not require a clean to the Point of Unification to ensure visibility by subsequent translation table walks.	

Bits	Name	Description	Reset
[19:16]	PAN	Privileged Access Never. Indicates support for the PAN bit in AArch32-CPSR, AArch32-SPSR, and AArch32-DSPSR in AArch32 state. Defined values are: **0b0010** PAN supported and AArch32-ATS1CPRP and AArch32-ATS1CPWP instructions supported.	
[15:12]	MaintBcst	Maintenance Broadcast. Indicates whether Cache, TLB, and branch predictor operations are broadcast. Defined values are: **0b0010** Cache, TLB, and branch predictor operations affect structures according to shareability and defined behavior of instructions.	
[11:8]	BPMaint	Branch Predictor Maintenance. Indicates the supported branch predictor maintenance operations in an implementation with hierarchical cache maintenance operations. Defined values are: **0b0010** Supported branch predictor maintenance operations are : Invalidate all branch predictors and invalidate branch predictors by Virtual Address (VA).	
[7:4]	CMaintSW	Cache Maintenance by Set/Way. Indicates the supported cache maintenance operations by set/way, in an implementation with hierarchical caches. Defined values are:	
[3:0]	CMaintVA	Cache Maintenance by Virtual Address. Indicates the supported cache maintenance operations by VA, in an implementation with hierarchical caches. Defined values are:	

Access

MRS <Xt>, ID_MMFR3_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_MMFR3_EL1	0b11	0b000	0b0000	0b0001	0b111

Accessibility

MRS <Xt>, ID_MMFR3_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_MMFR3_EL1;
elsif PSTATE.EL == EL2 then
    return ID_MMFR3_EL1;
elsif PSTATE.EL == EL3 then
    return ID_MMFR3_EL1;
```

B.5.12 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-66: AArch64_id_isar0_el1 bit assignments

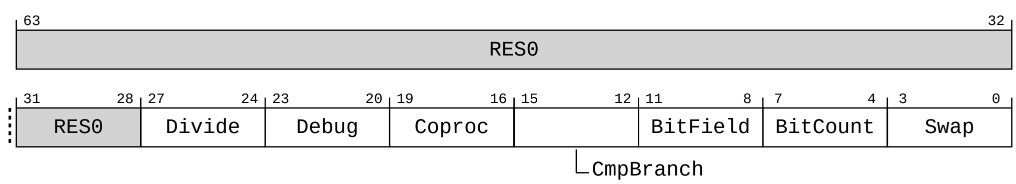


Table B-193: ID_ISAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:28]	RES0	Reserved	`0x0`
[27:24]	Divide	Indicates the implemented Divide instructions. **`0b0010`** Adds SDIV and UDIV in the T32 instruction set and adds SDIV and UDIV in the A32 instruction set.	
[23:20]	Debug	Indicates the implemented Debug instructions. Defined values are: **`0b0001`** Adds BKPT.	
[19:16]	Coproc	Indicates the implemented System register access instructions. Defined values are: **`0b0000`** None implemented, except for instructions separately attributed by the architecture to provide access to AArch32 System registers and System instructions.	
[15:12]	CmpBranch	Indicates the implemented combined Compare and Branch instructions in the T32 instruction set. Defined values are: **`0b0001`** Adds CBNZ and CBZ.	
[11:8]	BitField	Indicates the implemented BitField instructions. Defined values are: **`0b0001`** Adds BFC, BFI, SBFX, and UBFX.	

Bits	Name	Description	Reset
[7:4]	BitCount	Indicates the implemented Bit Counting instructions. Defined values are: **0b0001** Adds CLZ.	
[3:0]	Swap	Indicates the implemented Swap instructions in the A32 instruction set. Defined values are: **0b0000** None implemented.	

Access

MRS <Xt>, ID_ISAR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR0_EL1	0b11	0b000	0b0000	0b0010	0b000

Accessibility

MRS <Xt>, ID_ISAR0_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_ISAR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR0_EL1;
```

B.5.13 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-67: AArch64_id_isar1_el1 bit assignments

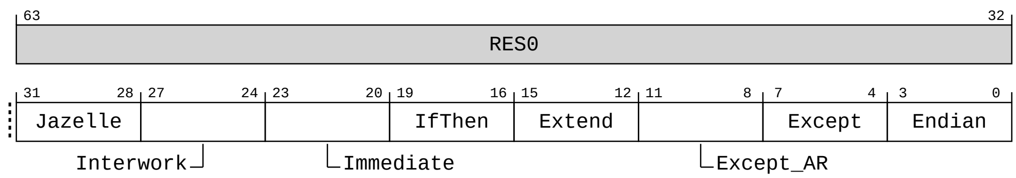


Table B-195: ID_ISAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	Jazelle	Indicates the implemented Jazelle extension instructions. Defined values are: **`0b0001`** Adds the BXJ instruction and the J bit in the PSR. This setting might indicate a trivial implementation of the Jazelle extension.	
[27:24]	Interwork	Indicates the implemented Interworking instructions. Defined values are: **`0b0011`** Adds the BX instruction, and the T bit in the PSR. Adds the BLX instruction and guarantees that data-processing instructions in the A32 instruction set with the PC as the destination and the S bit clear have BX-like behavior.	
[23:20]	Immediate	Indicates the implemented data-processing instructions with long immediates. Defined values are:	
[19:16]	IfThen	Indicates the implemented If-Then instructions in the T32 instruction set. Defined values are: **`0b0001`** Adds the IT instructions, and the IT bits in the PSRs.	
[15:12]	Extend	Indicates the implemented Extend instructions. Defined values are: **`0b0010`** Adds the SXTB, SXTH, UXTB, and UXTH. It also adds SXTB16, SXTAB, SXTAB16, SXTAH, UXTB16, UXTAB,UXTAB16, and UXTAH instructions	
[11:8]	Except_AR	Indicates the implemented A and R profile exception-handling instructions. Defined values are: **`0b0001`** Adds the SRS and RFE instructions, and the A and R profile forms of the CPS instruction.	
[7:4]	Except	Indicates the implemented exception-handling instructions in the A32 instruction set. Defined values are: **`0b0001`** Adds the LDM (exception return), LDM (user registers), and STM (user registers) instruction versions.	
[3:0]	Endian	Indicates the implemented Endian instructions. Defined values are: **`0b0001`** Adds the SETEND instruction, and the E bit in the PSRs.	

Access

MRS <Xt>, ID_ISAR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR1_EL1	0b11	0b000	0b0000	0b0010	0b001

Accessibility

MRS <Xt>, ID_ISAR1_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_ISAR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR1_EL1;
```

B.5.14 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-68: AArch64_id_isar2_el1 bit assignments

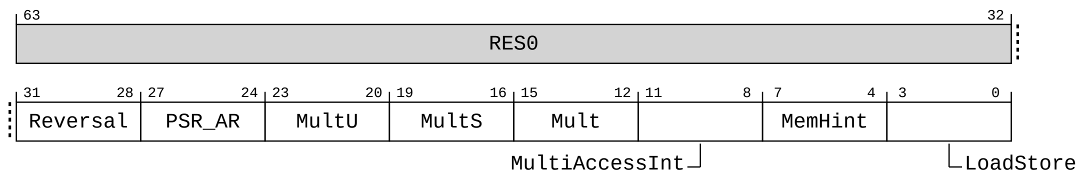


Table B-197: ID_ISAR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:28]	Reversal	Indicates the implemented Reversal instructions. Defined values are: **0b0010** Adds the REV, REV16, and REVSH and RBIT instructions	
[27:24]	PSR_AR	Indicates the implemented A and R profile instructions to manipulate the PSR. Defined values are: **0b0001** Adds the MRS and MSR instructions, and the exception return forms of data-processing instructions.	
[23:20]	MultU	Indicates the implemented advanced unsigned Multiply instructions. Defined values are: **0b0010** As for 0b0001, and adds the UMAAL instruction.	
[19:16]	MultS	Indicates the implemented advanced signed Multiply instructions. Defined values are: **0b0011** Adds the SMULL and SMLAL instructions. It addes the SMLABB, SMLABT, SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLATB, SMLATT, SMLAWB, SMLAWT, SMULBB, SMULBT, SMULTB, SMULTT, SMULWB, and SMULWT instructions. Also adds the Q bit in the PSRs. Adds the SMLAD, SMLADX, SMLALD, SMLALDX, SMLSD,SMLSDX, SMLSLD, SMLSLDX, SMMLA, SMMLAR, SMMLS, SMMLSR, SMMUL, SMMULR, SMUAD, SMUADX, SMUSD, and SMUSDX instructions	
[15:12]	Mult	Indicates the implemented additional Multiply instructions. Defined values are: **0b0010** Adds the MLA instruction+ and adds the MLS instruction	
[11:8]	MultiAccessInt	Indicates the support for interruptible multi-access instructions. Defined values are: **0b0000** No support. This means the LDM and STM instructions are not interruptible.	
[7:4]	MemHint	Indicates the implemented Memory Hint instructions. Defined values are: **0b0100** Adds the PLD, PLI and PLDW instruction	
[3:0]	LoadStore	Indicates the implemented additional load/store instructions. Defined values are: **0b0010** Adds the LDRD and STRD instructions and adds the Load Acquire (LDAB, LDAH, LDA, LDAEXB, LDAEXH, LDAEX, LDAEXD) and Store Release (STLB, STLH, STL, STLEXB, TLEXH, STLEX, STLEXD) instructions	

Access

MRS <Xt>, ID_ISAR2_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR2_EL1	0b11	0b000	0b0000	0b0010	0b010

Accessibility

MRS <Xt>, ID_ISAR2_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_ISAR2_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR2_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR2_EL1;
```

B.5.15 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-69: AArch64_id_isar3_el1 bit assignments

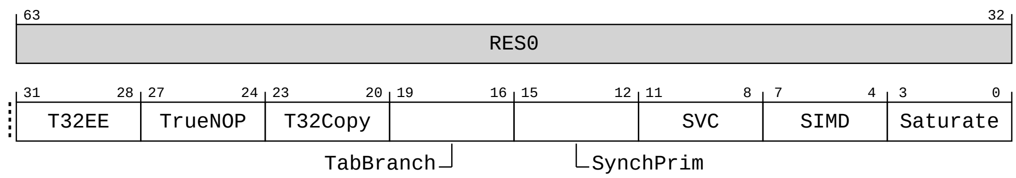


Table B-199: ID_ISAR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	T32EE	Indicates the implemented T32EE instructions. Defined values are: **`0b0000`** None implemented.	
[27:24]	TrueNOP	Indicates the implemented true NOP instructions. Defined values are: **`0b0001`** Adds true NOP instructions in both the T32 and A32 instruction sets. This also permits additional NOP-compatible hints.	
[23:20]	T32Copy	Indicates the support for T32 non flag-setting MOV instructions. Defined values are: **`0b0001`** Adds support for T32 instruction set encoding T1 of the MOV (register) instruction, copying from a low register to a low register.	
[19:16]	TabBranch	Indicates the implemented Table Branch instructions in the T32 instruction set. Defined values are: **`0b0001`** Adds the TBB and TBH instructions.	
[15:12]	SynchPrim	Used in conjunction with ID_ISAR4.SynchPrim_frac to indicate the implemented Synchronization Primitive instructions. Defined values are: **`0b0010`** Adds the LDREX and STREX, CLREX, LDREXB, STREXB, LDREXD and STREXD instructions.	
[11:8]	SVC	Indicates the implemented SVC instructions. Defined values are: **`0b0001`** Adds the SVC instruction.	
[7:4]	SIMD	Indicates the implemented SIMD instructions. Defined values are: **`0b0011`** Adds the SSAT and USAT instructions, and the Q bit in the PSRs. It also adds the PKHBT, PKHTB, QADD16, QADD8, QASX, QSUB16, QSUB8, QSAX, SADD16, SADD8, SASX, SEL, SHADD16, SHADD8, SHASX, SHSUB16, SHSUB8, SHSAX, SSAT16, SSUB16, SSUB8, SSAX, SXTAB16, SXTB16, UADD16, UADD8, UASX, UHADD16, UHADD8, UHASX, UHSUB16, UHSUB8, UHSAX, UQADD16, UQADD8, UQASX, UQSUB16, UQSUB8, UQSAX, USAD8, USADA8, USAT16, USUB16, USUB8, USAX, UXTAB16, and UXTB16 instructions. Also adds support for the GE[3:0] bits in the PSRs.	
[3:0]	Saturate	Indicates the implemented Saturate instructions. Defined values are: **`0b0001`** Adds the QADD, QDADD, QDSUB, and QSUB instructions, and the Q bit in the PSRs.	

Access

MRS <Xt>, ID_ISAR3_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR3_EL1	0b11	0b000	0b0000	0b0010	0b011

Accessibility

MRS <Xt>, ID_ISAR3_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_ISAR3_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR3_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR3_EL1;
```

B.5.16 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, and AArch64-ID_ISAR5_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-70: AArch64_id_isar4_el1 bit assignments

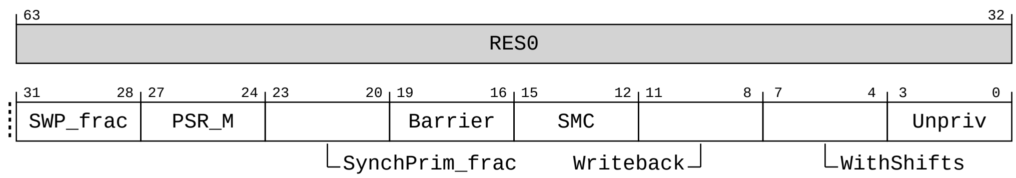


Table B-201: ID_ISAR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:28]	SWP_frac	Indicates support for the memory system locking the bus for SWP or SWPB instructions. Defined values are: **0b0000** SWP or SWPB instructions not implemented.	
[27:24]	PSR_M	Indicates the implemented M profile instructions to modify the PSRs. Defined values are: **0b0000** None implemented.	
[23:20]	SynchPrim_frac	Used in conjunction with AArch32-ID_ISAR3.SynchPrim to indicate the implemented Synchronization Primitive instructions. Possible values are: **0b0000** If SynchPrim == 0b0000, no Synchronization Primitives implemented. If SynchPrim == 0b0001, adds the LDREX and STREX instructions. If SynchPrim == 0b0010, also adds the CLREX, LDREXB, LDREXH, STREXB, STREXH, LDREXD, and STREXD instructions.	
[19:16]	Barrier	Indicates the implemented Barrier instructions in the A32 and T32 instruction sets. Defined values are: **0b0001** Adds the DMB, DSB, and ISB barrier instructions.	
[15:12]	SMC	Indicates the implemented SMC instructions. Defined values are: **0b0000** None implemented.	
[11:8]	Writeback	Indicates the support for Writeback addressing modes. Defined values are: **0b0001** Adds support for all of the writeback addressing modes.	
[7:4]	WithShifts	Indicates the support for instructions with shifts. Defined values are: **0b0100** Adds support for shifts of loads and stores over the range LSL 0-3. It adds support for other constant shift options, both on load/store and other instructions. It also adds support for register-controlled shift options.	
[3:0]	Unpriv	Indicates the implemented unprivileged instructions. Defined values are: **0b0010** Adds the LDRBT, LDRT, STRBT, and STRT instructions and adds the LDRHT, LDRSBT, LDRSHT, and STRHT instructions.	

Access

MRS <Xt>, ID_ISAR4_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR4_EL1	0b11	0b000	0b0000	0b0010	0b100

Accessibility

MRS <Xt>, ID_ISAR4_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_ISAR4_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR4_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR4_EL1;
```

B.5.17 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, and AArch64-ID_ISAR4_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-71: AArch64_id_isar5_el1 bit assignments

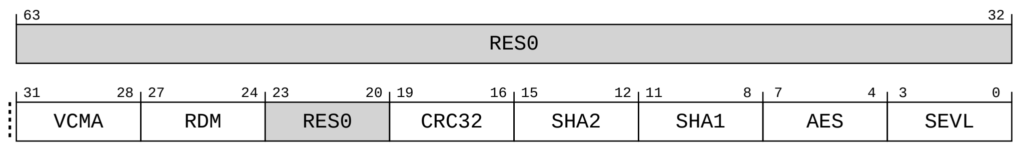


Table B-203: ID_ISAR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	VCMA	Indicates AArch32 support for complex number addition and multiplication where numbers are stored in vectors. Defined values are: **`0b0001`** The VCMLA and VCADD instructions are implemented in AArch32.	
[27:24]	RDM	Indicates whether the VQRDMLAH and VQRDMLSH instructions are implemented in AArch32 state. Defined values are: **`0b0001`** VQRDMLAH and VQRDMLSH instructions implemented.	
[23:20]	RES0	Reserved	`0b0000`
[19:16]	CRC32	Indicates whether the CRC32 instructions are implemented in AArch32 state. **`0b0001`** CRC32B, CRC32H, CRC32W, CRC32CB, CRC32CH, and CRC32CW instructions implemented.	
[15:12]	SHA2	Indicates whether the SHA2 instructions are implemented in AArch32 state. **`0b0000`** When Cryptographic extensions are not implemented or disabled then SHA2 instructions are not implemented. **`0b0001`** When Cryptographic extensions are implemented and enabled then SHA256H, SHA256H2, SHA256SU0, and SHA256SU1 instructions are implemented.	
[11:8]	SHA1	Indicates whether the SHA1 instructions are implemented in AArch32 state. **`0b0000`** When Cryptographic extensions are not implemented or disabled then SHA1 instructions are not implemented. **`0b0001`** When Cryptographic extensions are implemented and enabled then SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions are implemented.	

Bits	Name	Description	Reset
[7:4]	AES	Indicates whether the AES instructions are implemented in AArch32 state. **`0b0000`** When Cryptographic extensions are not implemented or disabled then AES instructions are not implemented. **`0b0010`** When Cryptographic extensions are implemented and enabled then AESE, AESD, AESMC, AESIMC and VMULL.64 instructions are implemented.	
[3:0]	SEVL	Indicates whether the SEVL instruction is implemented in AArch32 state. **`0b0001`** SEVL is implemented as Send Event Local.	

Access

MRS <Xt>, ID_ISAR5_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR5_EL1	`0b11`	`0b000`	`0b0000`	`0b0010`	`0b101`

Accessibility

MRS <Xt>, ID_ISAR5_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_ISAR5_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR5_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR5_EL1;
```

B.5.18 ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4

Provides information about the implemented memory model and memory management support in AArch32 state.

Must be interpreted with AArch64-ID_MMFR0_EL1, AArch64-ID_MMFR1_EL1, AArch64-ID_MMFR2_EL1, and AArch64-ID_MMFR3_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-72: AArch64_id_mmfr4_el1 bit assignments

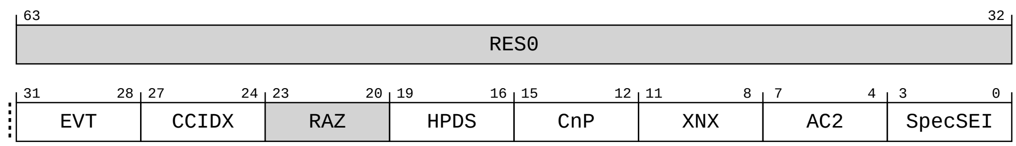


Table B-205: ID_MMFR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	EVT	Enhanced Virtualization Traps. If EL2 is implemented, indicates support for the AArch32-HCR2.{TTLBIS, TOCU, TICAB, TID4} traps. Defined values are: **`0b0010`** AArch32-HCR2.{TTLBIS, TOCU, TICAB, TID4} traps are supported.	
[27:24]	CCIDX	Support for use of the revised CCSIDR format and the presence of the CCSIDR2 is indicated. Defined values are: **`0b0001`** 64-bit format implemented for all levels of the CCSIDR, and the CCSIDR2 register is implemented.	
[23:20]	RAZ	Reserved	
[19:16]	HPDS	Hierarchical permission disables bits in translation tables. Defined values are: **`0b0010`** Supports disabling of hierarchical controls using the AArch32-TTBCR2.HPD0, AArch32-TTBCR2.HPD1, and AArch32-HTCR.HPD bits and adds possible hardware allocation of bits[62:59] of the translation table descriptors from the final lookup level for IMPLEMENTATION DEFINED usage	
[15:12]	CnP	Common not Private translations. Defined values are: **`0b0001`** Common not Private translations supported.	
[11:8]	XNX	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.	
[7:4]	AC2	Indicates the extension of the AArch32-ACTLR and AArch32-HACTLR registers using AArch32-ACTLR2 and rHACTLR2. Defined values are: **`0b0001`** AArch32-ACTLR2 and rHACTLR2 are implemented.	

Bits	Name	Description	Reset
[3:0]	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.	

Access

MRS <Xt>, ID_MMFR4_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_MMFR4_EL1	`0b11`	`0b000`	`0b0000`	`0b0010`	`0b110`

Accessibility

MRS <Xt>, ID_MMFR4_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() && (!IsZero(ID_MMFR4_EL1) || boolean IMPLEMENTATION_DEFINED
 "ID_MMFR4_EL1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_MMFR4_EL1;
elsif PSTATE.EL == EL2 then
    return ID_MMFR4_EL1;
elsif PSTATE.EL == EL3 then
    return ID_MMFR4_EL1;
```

B.5.19 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1 and AArch64-ID_ISAR5_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-73: AArch64_id_isar6_el1 bit assignments

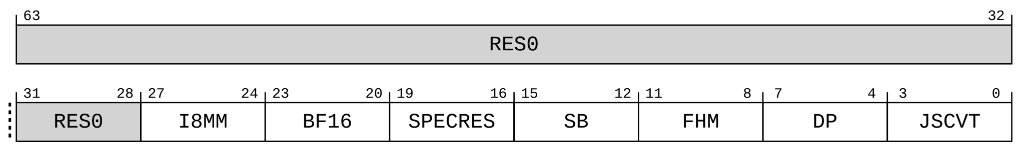


Table B-207: ID_ISAR6_EL1 bit descriptions

Bits	Name	Description	Reset
[63:28]	RES0	Reserved	`0x0`
[27:24]	I8MM	Indicates support for Advanced SIMD and floating-point Int8 matrix multiplication instructions in AArch32 state. Defined values of this field are: **`0b0001`** VSMMLA, VSUDOT, VUMMLA, VUSMMLA, and VUSDOT instructions are implemented.	
[23:20]	BF16	Indicates support for Advanced SIMD and floating-point BFloat16 instructions in AArch32 state. Defined values are: **`0b0001`** VCVT, VCVTB, VCVTT, VDOT, VFMAL, and VMMLA instructions with BF16 operand or result types are implemented.	
[19:16]	SPECRES	Indicates support for Speculation invalidation instructions in AArch32 state. Defined values are: **`0b0001`** CFPRCTX, DVPRCTX, and CPPRCTX instructions are implemented.	
[15:12]	SB	Indicates support for the SB instruction in AArch32 state. Defined values are: **`0b0001`** SB instruction is implemented.	
[11:8]	FHM	Indicates support for Advanced SIMD and floating-point VFMAL and VFMSL instructions in AArch32 state. Defined values are: **`0b0001`** VFMAL and VMFSL instructions are implemented.	
[7:4]	DP	Indicates support for Advanced SIMD and floating-point VFMAL and VFMSL instructions in AArch32 state. Defined values are: **`0b0001`** UDOT and VSDOT instructions are implemented.	

Bits	Name	Description	Reset
[3:0]	JSCVT	Indicates support for the VJCVT instruction in AArch32 state. Defined values are: **0b0001** The VJCVT instruction is implemented.	

Access

MRS <Xt>, ID_ISAR6_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR6_EL1	0b11	0b000	0b0000	0b0010	0b111

Accessibility

MRS <Xt>, ID_ISAR6_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() && (!IsZero(ID_ISAR6_EL1) || boolean IMPLEMENTATION_DEFINED
 "ID_ISAR6_EL1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_ISAR6_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR6_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR6_EL1;
```

B.5.20 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'.

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

Reset value

See individual bit resets.

Bit descriptions

Figure B-74: AArch64_mvfr0_el1 bit assignments

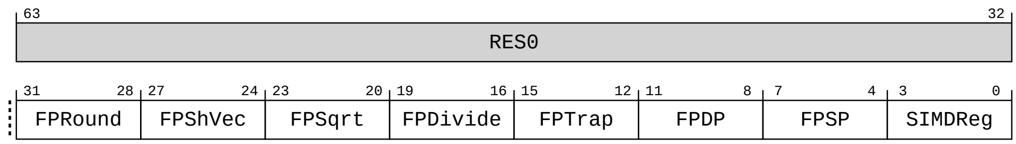


Table B-209: MVFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	FPRound	Floating-Point Rounding modes. Indicates whether the floating-point implementation provides support for rounding modes. Defined values are: **`0b0001`** All rounding modes supported.	
[27:24]	FPShVec	Short Vectors. Indicates whether the floating-point implementation provides support for the use of short vectors. Defined values are: **`0b0000`** Short vectors not supported.	
[23:20]	FPSqrt	Square Root. Indicates whether the floating-point implementation provides support for the ARMv6 VFP square root operations. Defined values are: **`0b0001`** Supported.	
[19:16]	FPDivide	Indicates whether the floating-point implementation provides support for VFP divide operations. Defined values are: **`0b0001`** Supported.	
[15:12]	FPTrap	Floating Point Exception Trapping. Indicates whether the floating-point implementation provides support for exception trapping. Defined values are: **`0b0000`** Not supported.	
[11:8]	FPDP	Double Precision. Indicates whether the floating-point implementation provides support for double-precision operations. Defined values are: **`0b0010`** Supported, VFPv3, VFPv4, or Armv8. VFPv3 and Armv8 add an instruction to load a double-precision floating-point constant, and conversions between double-precision and fixed-point values.	
[7:4]	FPSP	Single Precision. Indicates whether the floating-point implementation provides support for single-precision operations. Defined values are: **`0b0010`** Supported, VFPv3 or VFPv4. VFPv3 adds an instruction to load a single-precision floating-point constant, and conversions between single-precision and fixed-point values.	

Bits	Name	Description	Reset
[3:0]	SIMDReg	Advanced SIMD registers. Indicates whether the Advanced SIMD and floating-point implementation provides support for the Advanced SIMD and floating-point register bank. Defined values are: **`0b0010`** The implementation includes Advanced SIMD and floating-point support with 32 x 64-bit registers.	

Access

MRS <Xt>, MVFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MVFR0_EL1	`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;
```

B.5.21 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'.

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

Reset value

See individual bit resets.

Bit descriptions

Figure B-75: AArch64_mvfr1_el1 bit assignments

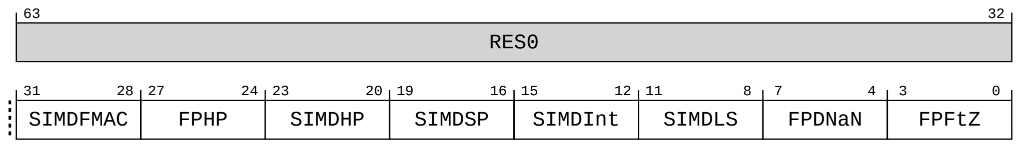


Table B-211: MVFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	SIMDFMAC	Advanced SIMD Fused Multiply-Accumulate. Indicates whether the Advanced SIMD implementation provides fused multiply accumulate instructions. Defined values are: **`0b0001`** Implemented.	
[27:24]	FPHP	Floating Point Half Precision. Indicates the level of half-precision floating-point support. Defined values are: **`0b0011`** As for 0b0010, and adds support for half-precision floating-point arithmetic.	
[23:20]	SIMDHP	Advanced SIMD Half Precision. Indicates the level of half-precision floating-point support. Defined values are: **`0b0010`** As for 0b0001, and adds support for half-precision floating-point arithmetic.	
[19:16]	SIMDSP	Advanced SIMD Single Precision. Indicates whether the Advanced SIMD and floating-point implementation provides single-precision floating-point instructions. Defined values are: **`0b0001`** Implemented. This value is permitted only if the SIMDInt field is 0b0001.	
[15:12]	SIMDInt	Advanced SIMD Integer. Indicates whether the Advanced SIMD and floating-point implementation provides integer instructions. Defined values are: **`0b0001`** Implemented.	
[11:8]	SIMDLS	Advanced SIMD Load/Store. Indicates whether the Advanced SIMD and floating-point implementation provides load/store instructions. Defined values are: **`0b0001`** Implemented.	
[7:4]	FPDNaN	Default NaN mode. Indicates whether the floating-point implementation provides support only for the Default NaN mode. Defined values are: **`0b0001`** Hardware supports propagation of NaN values.	
[3:0]	FPFtZ	Flush to Zero mode. Indicates whether the floating-point implementation provides support only for the Flush-to-Zero mode of operation. Defined values are: **`0b0001`** Hardware supports full denormalized number arithmetic.	

Access

MRS <Xt>, MVFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MVFR1_EL1	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;
```

B.5.22 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'.

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

Reset value

See individual bit resets.

Bit descriptions

Figure B-76: AArch64_mvfr2_el1 bit assignments

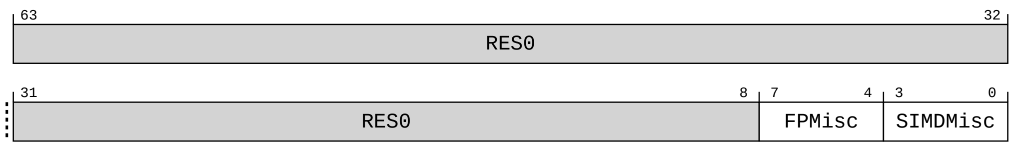


Table B-213: MVFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	`0x0`
[7:4]	FPMisc	Indicates whether the floating-point implementation provides support for miscellaneous VFP features. **`0b0100`** As 0b0011, and Floating-point MaxNum and MinNum.	
[3:0]	SIMDMisc	Indicates whether the Advanced SIMD implementation provides support for miscellaneous Advanced SIMD features. **`0b0011`** As 0b0010, and Floating-point MaxNum and MinNum.	

Access

MRS <Xt>, MVFR2_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MVFR2_EL1	`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;
```

B.5.23 ID_PFR2_EL1, AArch32 Processor Feature Register 2

Gives information about the AArch32 programmers' model.

Must be interpreted with AArch64-ID_PFR0_EL1 and AArch64-ID_PFR1_EL1. For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-77: AArch64_id_pfr2_el1 bit assignments

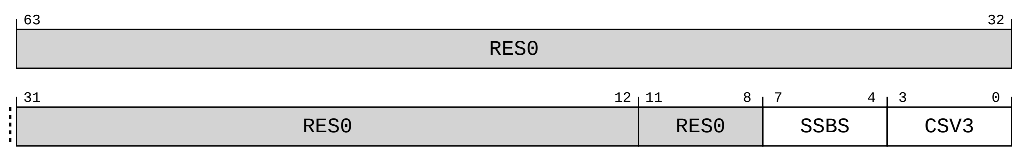


Table B-215: ID_PFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	`0b0000`
[7:4]	SSBS	Speculative Store Bypassing controls in AArch64 state. Defined values are: **`0b0001`** AArch32 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypass Safe.	
[3:0]	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	

Access

MRS <Xt>, ID_PFR2_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_PFR2_EL1	0b11	0b000	0b0000	0b0011	0b100

Accessibility

MRS <Xt>, ID_PFR2_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_PFR2_EL1;
elsif PSTATE.EL == EL2 then
    return ID_PFR2_EL1;
elsif PSTATE.EL == EL3 then
    return ID_PFR2_EL1;
```

B.5.24 ID_DFR1_EL1, Debug Feature Register 1

Provides top level information about the debug system in AArch32.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

identification

Reset value

0x0

Bit descriptions

Figure B-78: AArch64_id_dfr1_el1 bit assignments

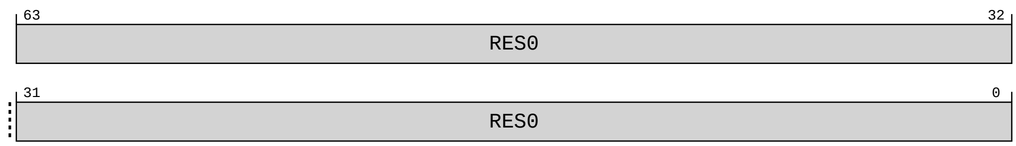


Table B-217: ID_DFR1_EL1 bit descriptions

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

Access

MRS <Xt>, ID_DFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_DFR1_EL1	0b11	0b000	0b0000	0b0011	0b101

Accessibility

MRS <Xt>, ID_DFR1_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() && (!IsZero(ID_DFR1_EL1) || boolean IMPLEMENTATION_DEFINED
 "ID_DFR1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_DFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_DFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_DFR1_EL1;
```

B.5.25 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0

Provides additional information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

The external register ext-EDPFR gives information from this register.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-79: AArch64_id_aa64pfr0_el1 bit assignments

63:60	59:56	55:52	51:48	47:44	43:40	39:36	35:32
CSV3	CSV2	RES0	DIT	AMU	MPAM	SEL2	SVE

31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
RAS	GIC	AdvSIMD	FP	EL3	EL2	EL1	EL0

Table B-219: ID_AA64PFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	CSV3	Speculative use of faulting data. Defined values are: **`0b0001`** Data loaded under speculation with a permission or domain fault cannot be used to form an address or generate condition codes or SVE predicate values to be used by instructions newer than the load in the speculative sequence	
[59:56]	CSV2	Speculative use of out of context branch targets. Defined values are: **`0b0010`** Branch targets trained in one hardware described context can only affect speculative execution in a different hardware described context in a hard-to-determine way. Contexts include the SCXTNUM_ELx register contexts, and these registers are supported.	
[55:52]	RES0	Reserved	`0b0000`
[51:48]	DIT	Data Independent Timing. Defined values are: **`0b0001`** AArch64 provides the PSTATE.DIT mechanism to guarantee constant execution time of certain instructions.	
[47:44]	AMU	Indicates support for Activity Monitors Extension. Defined values are: **`0b0001`** AMUv1 for Armv8.4 is implemented.	
[43:40]	MPAM	Indicates support for MPAM Extension. Defined values are: **`0b0001`** If AArch64-ID_AA64PFR1_EL1.MPAM_frac == 0b0000, MPAM Extension version 1.0 is implemented. If AArch64-ID_AA64PFR1_EL1.MPAM_frac == 0b0001, MPAM Extension version 1.1 is implemented.	

Bits	Name	Description	Reset
[39:36]	SEL2	Secure EL2. Defined values are: **0b0001** Secure EL2 is implemented.	
[35:32]	SVE	Scalable Vector Extension. Defined values are: **0b0001** SVE architectural state and programmers' model are implemented.	
[31:28]	RAS	RAS Extension version. Defined values are: **0b0010** ARMv8.4-RAS present. As 0b0001, and adds support for ARMv8.4-DFE (If EL3 is implemented), additional ERXMISCm_EL1 System registers, additionalSystem registers ERXPFGCDN_EL1, ERXPFGCTL_EL1, and ERXPFGF_EL1, and the SCR_EL3.FIEN and HCR_EL2.FIEN trap controls, to support the optional RAS Common Fault Injection Model Extension.	
[27:24]	GIC	System register GIC CPU interface. Defined values are: **0b0000** When Port GICCDISABLE is High, GIC CPU interface is disabled. **0b0011** When Port GICCDISABLE is Low, GIC (version 4.1) CPU interface is enabled.	
[23:20]	AdvSIMD	Advanced SIMD. Defined values are: **0b0001** Advanced SIMD is implemented, including support for half-precision floating-point arithmetic.	
[19:16]	FP	Floating-point. Defined values are: **0b0001** Floating-point, including support for half-precision floating-point arithmetic, is implemented.	
[15:12]	EL3	EL3 Exception level handling. Defined values are: **0b0001** EL3 can be executed in AArch64 state only.	
[11:8]	EL2	EL2 Exception level handling. Defined values are: **0b0001** EL2 can be executed in AArch64 state only.	
[7:4]	EL1	EL1 Exception level handling. Defined values are: **0b0001** EL1 can be executed in AArch64 state only.	
[3:0]	EL0	EL0 Exception level handling. Defined values are: **0b0010** EL0 can be executed in either AArch64 or AArch32 state.	

Access

MRS <Xt>, ID_AA64PFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64PFR0_EL1	0b11	0b000	0b0000	0b0100	0b000

Accessibility

MRS <Xt>, ID_AA64PFR0_EL1

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

B.5.26 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1

Reserved for future expansion of information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-80: AArch64_id_aa64pfr1_el1 bit assignments

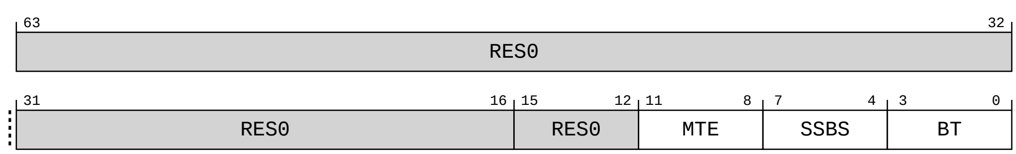


Table B-221: ID_AA64PFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	0b0000

Bits	Name	Description	Reset
[11:8]	MTE	Support for the Memory Tagging Extension. Defined values are: **0b0001** Memory Tagging Extension instructions accessible at EL0 are implemented. Instructions and System Registers defined by the extension not configurably accessible at EL0 are Unallocated and other System Register fields defined by the extension are RES0. This value is reported when the BROADCASTMTE input is LOW. **0b0010** Memory Tagging Extension is implemented. This value is reported when the BROADCASTMTE input is HIGH.	
[7:4]	SSBS	Speculative Store Bypassing controls in AArch64 state. Defined values are: **0b0010** AArch64 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypassing Safe, and the MSR and MRS instructions to directly read and write the PSTATE.SSBS field	
[3:0]	BT	Branch Target Identification mechanism support in AArch64 state. Defined values are: **0b0001** The Branch Target Identification mechanism is implemented.	

Access

MRS <Xt>, ID_AA64PFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64PFR1_EL1	0b11	0b000	0b0000	0b0100	0b001

Accessibility

MRS <Xt>, ID_AA64PFR1_EL1

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

B.5.27 ID_AA64ZFR0_EL1, SVE Feature ID register 0

Provides additional information about the implemented features of the AArch64 Scalable Vector Extension, when the AArch64-ID_AA64PFR0_EL1.SVE field is not zero.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-81: AArch64_id_aa64zfr0_el1 bit assignments

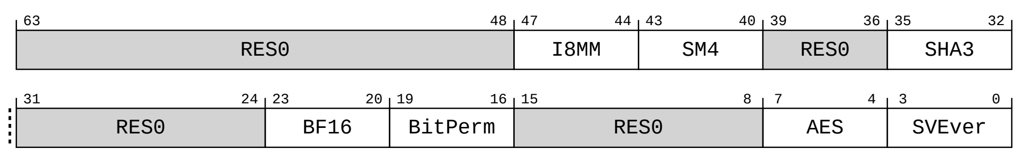


Table B-223: ID_AA64ZFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	`0x0`
[47:44]	I8MM	Indicates support for SVE Int8 matrix multiplication instructions. Defined values are: **`0b0001`** SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	
[43:40]	SM4	Indicates support for SVE2 SM4 instructions. Defined values are: **`0b0000`** SVE2 SM4 instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **`0b0001`** SVE2 SM4E and SM4EKEY instructions are implemented. This value is reported when Cryptographic extensions are implemented and enabled.	
[39:36]	RES0	Reserved	`0b0000`

Bits	Name	Description	Reset
[35:32]	SHA3	Indicates support for the SVE2 SHA-3 instruction. Defined values are: **0b0000** SVE2 SHA-3 instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SVE2 RAX1 instruction is implemented. This value is reported when Cryptographic extensions are implemented and enabled.	
[31:24]	RES0	Reserved	0b00000000
[23:20]	BF16	Indicates support for SVE BFloat16 instructions. Defined values are: **0b0001** BFCVT, BFCVTNT, BFDOT, BFMLALB, BFMLALT, and BFMMLA instructions are implemented.	
[19:16]	BitPerm	Indicates support for SVE2 bit permute instructions. Defined values are: **0b0001** SVE2 BDEP, BEXT and BGRP instructions are implemented.	
[15:8]	RES0	Reserved	0b00000000
[7:4]	AES	Indicates support for SVE2-AES instructions. Defined values are: **0b0000** SVE2-AES instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0010** SVE2 AESE, AESD, AESMC, and AESIMC instructions are implemented plus SVE2 PMULLB and PMULLT instructions with 64-bit source. This value is reported when Cryptographic extensions are implemented and enabled.	
[3:0]	SVEver	Scalable Vector Extension instruction set version. Defined values are: **0b0001** SVE and the non-optional SVE2 instructions are implemented.	

Access

MRS <Xt>, ID_AA64ZFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64ZFR0_EL1	0b11	0b000	0b0000	0b0100	0b100

Accessibility

MRS <Xt>, ID_AA64ZFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && (!IsZero(ID_AA64ZFR0_EL1) || boolean IMPLEMENTATION_DEFINED
 "ID_AA64ZFR0_EL1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
```

```
        return ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ZFR0_EL1;
```

B.5.28 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0

Provides top level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers, see Principles of the ID scheme for fields in ID registers.

Configurations

The external register ext-EDDFR gives information from this register.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-82: AArch64_id_aa64dfr0_el1 bit assignments

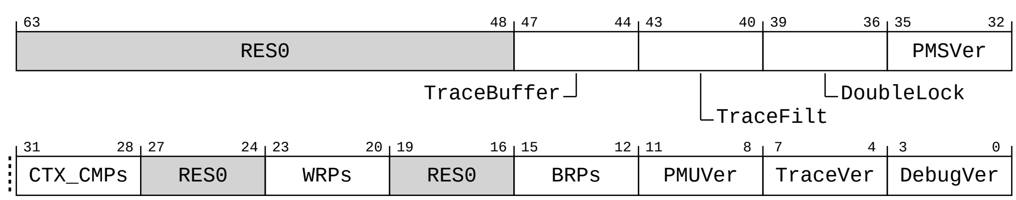


Table B-225: ID_AA64DFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	`0x0`
[47:44]	TraceBuffer	Trace Buffer Extension version. Defined values are: **`0b0001`** Trace Buffer Extension implemented.	
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. Defined values are: **`0b0001`** Armv8.4 Self-hosted Trace Extension implemented.	

Bits	Name	Description	Reset
[39:36]	DoubleLock	OS Double Lock implemented. Defined values are: **0b1111** OS Double Lock not implemented. AArch64-OSDLR_EL1 is RAZ/WI.	
[35:32]	PMSVer	Statistical Profiling Extension version. Defined values are:	
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. **0b0001** Two context-aware breakpoints are included	
[27:24]	RES0	Reserved	0b0000
[23:20]	WRPs	Number of watchpoints, minus 1. The value of 0000 is reserved. **0b0011** Four Watchpoints	
[19:16]	RES0	Reserved	0b0000
[15:12]	BRPs	Number of breakpoints, minus 1. The value of 0000 is reserved. **0b0101** Six Breakpoints	
[11:8]	PMUVer	Performance Monitors Extension version. Defined value is: **0b0110** Performance Monitors Extension implemented, PMUv3 for Armv8.5	
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a PE trace unit is implemented. Defined values are: **0b0001** PE trace unit System registers implemented.	
[3:0]	DebugVer	Debug architecture version. Indicates presence of Armv8 debug architecture. Defined values are: **0b1001** Armv8.4 debug architecture.	

Access

MRS <Xt>, ID_AA64DFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64DFR0_EL1	0b11	0b000	0b0000	0b0101	0b000

Accessibility

MRS <Xt>, ID_AA64DFR0_EL1

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

B.5.29 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1

Reserved for future expansion of top level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

0x0

Bit descriptions

Figure B-83: AArch64_id_aa64dfr1_el1 bit assignments

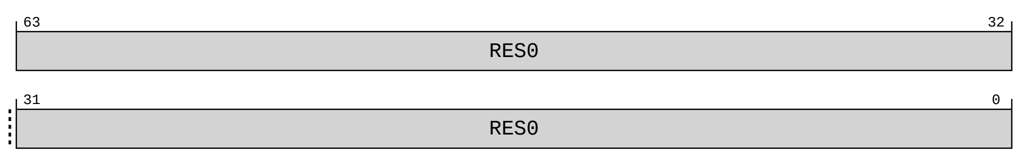


Table B-227: ID_AA64DFR1_EL1 bit descriptions

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

Access

MRS <Xt>, ID_AA64DFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64DFR1_EL1	0b11	0b000	0b0000	0b0101	0b001

Accessibility

MRS <Xt>, ID_AA64DFR1_EL1

```
if PSTATE.EL == EL0 then
```

```
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
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;
```

B.5.30 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0

Provides information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

0x0

Bit descriptions

Figure B-84: AArch64_id_aa64afr0_el1 bit assignments

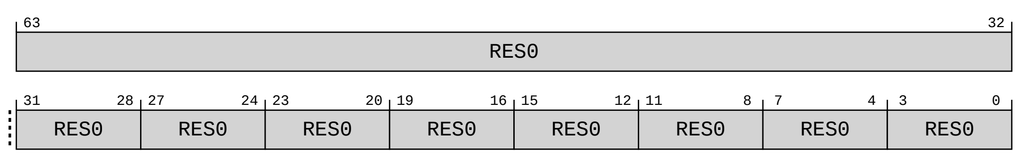


Table B-229: ID_AA64AFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0000

Access

MRS <Xt>, ID_AA64AFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64AFR0_EL1	0b11	0b000	0b0000	0b0101	0b100

Accessibility

MRS <Xt>, ID_AA64AFR0_EL1

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

B.5.31 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1

Reserved for future expansion of information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

0x0

Bit descriptions

Figure B-85: AArch64_id_aa64afr1_el1 bit assignments

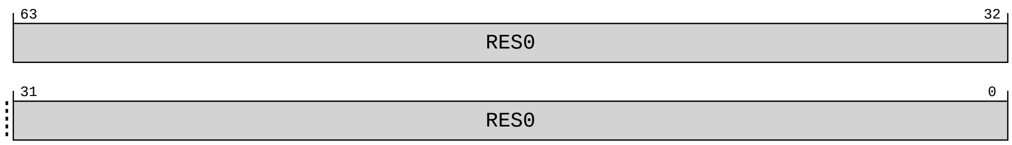


Table B-231: ID_AA64AFR1_EL1 bit descriptions

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

Access

MRS <Xt>, ID_AA64AFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64AFR1_EL1	`0b11`	`0b000`	`0b0000`	`0b0101`	`0b101`

Accessibility

MRS <Xt>, ID_AA64AFR1_EL1

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

B.5.32 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0

Provides information about the instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-86: AArch64_id_aa64isar0_el1 bit assignments

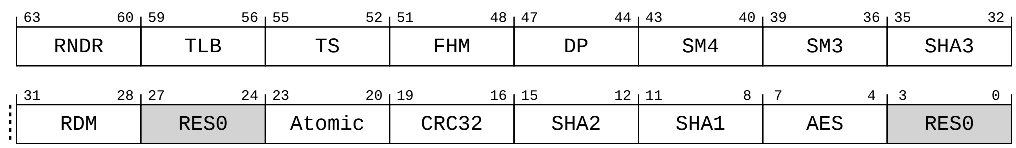


Table B-233: ID_AA64ISAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	RNDR	Indicates support for Random Number instructions in AArch64 state. Defined values are: **0b0000** No Random Number instructions are implemented. **0b0001** AArch64-RNDR and AArch64-RNDRRS registers are implemented, if the core has the RNDR feature configured.	
[59:56]	TLB	Indicates support for Outer shareable and TLB range maintenance instructions. Defined values are: **0b0010** Outer shareable and TLB range maintenance instructions are implemented.	
[55:52]	TS	Indicates support for flag manipulation instructions. Defined values are: **0b0010** CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented.	
[51:48]	FHM	Indicates support for FMLAL and FMLSL instructions. Defined values are: **0b0001** FMLAL and FMLSL instructions are implemented.	
[47:44]	DP	Indicates support for Dot Product instructions in AArch64 state. Defined values are: **0b0001** UDOT and SDOT instructions implemented.	

Bits	Name	Description	Reset
[43:40]	SM4	Indicates support for SM4 instructions in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then SM3 instructions are not implemented. **0b0001** When Cryptographic extensions are implemented and enabled then SM3 instructions SM4E and SM4EKEY are implemented.	
[39:36]	SM3	Indicates support for SM3 instructions in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then SM4 instructions are not implemented. **0b0001** When Cryptographic extensions are implemented and enabled then SM4 instructions SM3SS1, SM3TT1A, SM3TT1B, SM3TT2A, SM3TT2B, SM3PARTW1, and SM3PARTW2 are implemented.	
[35:32]	SHA3	Indicates support for SHA3 instructions in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then SHA3 instructions are not implemented. **0b0001** When Cryptographic extensions are implemented and enabled then SHA3 instructions EOR3, RAX1, XAR, and BCAX are implemented.	
[31:28]	RDM	Indicates support for SQRDMLAH and SQRDMLSH instructions in AArch64 state. Defined values are: **0b0001** SQRDMLAH and SQRDMLSH instructions implemented.	
[27:24]	RES0	Reserved	0b0000
[23:20]	Atomic	Indicates support for Atomic instructions in AArch64 state. Defined values are: **0b0010** LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions implemented.	
[19:16]	CRC32	CRC32 instructions implemented in AArch64 state. Defined values are: **0b0001** CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX instructions implemented.	
[15:12]	SHA2	SHA2 instructions implemented in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then SHA2 instructions are not implemented. **0b0010** When Cryptographic extensions are implemented and enabled then SHA256H, SHA256H2, SHA256SU0, SHA256SU1, SHA512H, SHA512H2, SHA512SU0, and SHA512SU1 instructions are implemented. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	

Bits	Name	Description	Reset
[11:8]	SHA1	SHA1 instructions implemented in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then SHA1 instructions are not implemented. **0b0001** When Cryptographic extensions are implemented and enabled then SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions are implemented. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	
[7:4]	AES	AES instructions implemented in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then AES instructions are not implemented. **0b0010** When Cryptographic extensions are implemented and enabled then AESE, AESD, AESMC, and AESIMC instructions are implemented and also PMULL/PMULL2 instructions operating on 64-bit data quantities. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	
[3:0]	RES0	Reserved	0b0000

Access

MRS <Xt>, ID_AA64ISAR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64ISAR0_EL1	0b11	0b000	0b0000	0b0110	0b000

Accessibility

MRS <Xt>, ID_AA64ISAR0_EL1

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

B.5.33 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

If ID_AA64ISAR1_EL1.{API, APA} == {0000, 0000}, then:

- The AArch64-TCR_EL1.{TBID,TBID0}, AArch64-TCR_EL2.{TBID0,TBID1}, AArch64-TCR_EL2.TBID and AArch64-TCR_EL3.TBID bits are RES0.
- AArch64-APIAKeyHi_EL1, AArch64-APIAKeyLo_EL1, AArch64-APIBKeyHi_EL1, AArch64-APIBKeyLo_EL1, AArch64-APDAKeyHi_EL1, AArch64-APDAKeyLo_EL1, AArch64-APDBKeyHi_EL1, AArch64-APDBKeyLo_EL1 are not allocated.
- 'SCTLR_EL'.EnIA, 'SCTLR_EL'.EnIB, 'SCTLR_EL'.EnDA, 'SCTLR_EL'.EnDB are all RES0.

If ID_AA64ISAR1_EL1.{GPI, GPA, API, APA} == {0000, 0000, 0000, 0000}, then:

- AArch64-HCR_EL2.APK and AArch64-HCR_EL2.API are RES0.
- AArch64-SCR_EL3.APK and AArch64-SCR_EL3.API are RES0.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-87: AArch64_id_aa64isar1_el1 bit assignments

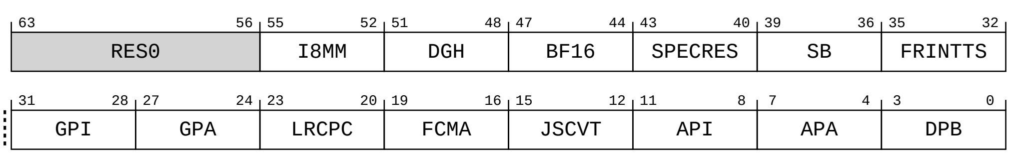


Table B-235: ID_AA64ISAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	0b00000000

Bits	Name	Description	Reset
[55:52]	I8MM	Indicates support for Advanced SIMD and Floating-point Int8 matrix multiplication instructions in AArch64 state. Defined values of this field are: **`0b0001`** SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	
[51:48]	DGH	Indicates support for the Data Gathering Hint instruction. Defined values are: **`0b0001`** Data Gathering Hint is implemented.	
[47:44]	BF16	Indicates support for Advanced SIMD and Floating-point BFloat16 instructions in AArch64 state. Defined values are: **`0b0001`** BFDOT, BFMLAL, BFMLAL2, BFMMLA, BFCVT, and BFCVT2 instructions are implemented.	
[43:40]	SPECRES	Indicates support for prediction invalidation instructions in AArch64 state. Defined values are: **`0b0001`** CFP RCTX, DVP RCTX, and CPP RCTX instructions are implemented.	
[39:36]	SB	Indicates support for SB instruction in AArch64 state. Defined values are: **`0b0001`** SB instruction is implemented.	
[35:32]	FRINTTS	Indicates support for the FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented. Defined values are: **`0b0001`** FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented.	
[31:28]	GPI	Indicates support for an IMPLEMENTATION DEFINED algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: **`0b0000`** Generic Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented.	
[27:24]	GPA	Indicates whether QARMA or Architected algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: **`0b0001`** Generic Authentication using the QARMA algorithm is implemented. This includes the PACGA instruction.	
[23:20]	LRCPC	Indicates support for weaker release consistency, RCpc, based model. Defined values are: **`0b0010`** The LDAPR*, LDAPUR*, and STLUR* instructions are implemented.	
[19:16]	FCMA	Indicates support for complex number addition and multiplication, where numbers are stored in vectors. Defined values are: **`0b0001`** The FCMLA and FCADD instructions are implemented.	
[15:12]	JSCVT	Indicates support for JavaScript conversion from double precision floating point values to integers in AArch64 state. Defined values are: **`0b0001`** The FJCVTZS instruction is implemented.	

Bits	Name	Description	Reset
[11:8]	API	Indicates whether an **IMPLEMENTATION DEFINED** algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are: `0b0000` Address Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented.	
[7:4]	APA	Indicates whether QARMA or Architected algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are: `0b0101` Address Authentication using the QARMA algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.	
[3:0]	DPB	Data Persistence writeback. Indicates support for the rDC CVAP and rDC CVADP instructions in AArch64 state. Defined values are: `0b0010` DC CVAP and DC CVADP supported	

Access

MRS <Xt>, ID_AA64ISAR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64ISAR1_EL1	0b11	0b000	0b0000	0b0110	0b001

Accessibility

MRS <Xt>, ID_AA64ISAR1_EL1

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

B.5.34 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-88: AArch64_id_aa64mmfr0_el1 bit assignments

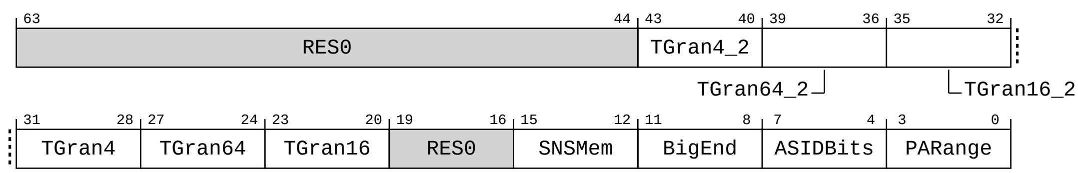


Table B-237: ID_AA64MMFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:44]	RES0	Reserved	`0x0`
[43:40]	TGran4_2	Indicates support for 4KB memory granule size for stage 2. Defined values are: **`0b0010`** 4KB granule supported at stage 2.	
[39:36]	TGran64_2	Indicates support for 64KB memory granule size for stage 2. Defined values are: **`0b0010`** 64KB granule supported at stage 2.	
[35:32]	TGran16_2	Indicates support for 16KB memory granule size for stage 2. Defined values are: **`0b0010`** 16KB granule supported at stage 2	
[31:28]	TGran4	Indicates support for 4KB memory translation granule size. Defined values are: **`0b0000`** 4KB granule supported.	
[27:24]	TGran64	Indicates support for 64KB memory translation granule size. Defined values are: **`0b0000`** 64KB granule supported.	
[23:20]	TGran16	Indicates support for 16KB memory translation granule size. Defined values are: **`0b0001`** 16KB granule supported.	
[19:16]	RES0	Reserved	`0b0000`

Bits	Name	Description	Reset
[15:12]	SNSMem	Indicates support for a distinction between Secure and Non-secure Memory. Defined values are: **`0b0001`** Does support a distinction between Secure and Non-secure Memory.	
[11:8]	BigEnd	Indicates support for mixed-endian configuration. Defined values are: **`0b0001`** Mixed-endian support. The SCTLR_ELx.EE and SCTLR_EL1.E0E bits can be configured.	
[7:4]	ASIDBits	Number of ASID bits. Defined values are: **`0b0010`** 16 bits.	
[3:0]	PARange	Physical Address range supported. Defined values are: **`0b0101`** 48 bits, 256TB.	

Access

MRS <Xt>, ID_AA64MMFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64MMFR0_EL1	`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;
```

B.5.35 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-89: AArch64_id_aa64mmfr1_el1 bit assignments

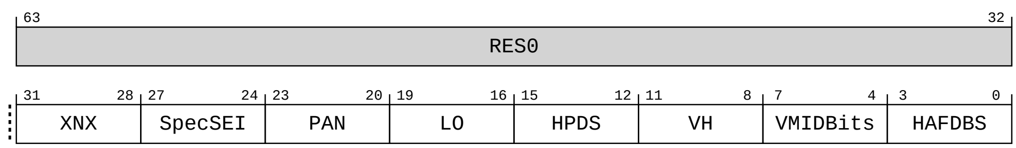


Table B-239: ID_AA64MMFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	XNX	Indicates support for execute-never control distinction by Exception level at stage 2. Defined values are: **`0b0001`** Distinction between EL0 and EL1 execute-never control at stage 2 supported.	
[27:24]	SpecSEI	Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches. The defined values of this field are: **`0b0000`** The PE never generates an SError interrupt due to an External abort on a speculative read.	
[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: **`0b0010`** PAN supported and rAT S1E1RP and rAT S1E1WP instructions supported.	
[19:16]	LO	LORegions. Indicates support for LORegions. Defined values are: **`0b0001`** LORegions supported.	
[15:12]	HPDS	Hierarchical Permission Disables. Indicates support for disabling hierarchical controls in translation tables. Defined values are: **`0b0010`** Disabling of hierarchical controls supported with the TCR_EL1.{HPD1, HPD0}, TCR_EL2.HPD or TCR_EL2.{HPD1, HPD0}, and TCR_EL3.HPD bits and adds possible hardware allocation of bits[62:59] of the translation table descriptors from the final lookup level for IMPLEMENTATION DEFINED use.	
[11:8]	VH	Virtualization Host Extensions. Defined values are: **`0b0001`** Virtualization Host Extensions supported.	

Bits	Name	Description	Reset
[7:4]	VMIDBits	Number of VMID bits. Defined values are: **0b0010** 16 bits	
[3:0]	HAFDBS	Hardware updates to Access flag and Dirty state in translation tables. Defined values are: **0b0010** Hardware update of both the Access flag and dirty state is supported.	

Access

MRS <Xt>, ID_AA64MMFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64MMFR1_EL1	0b11	0b000	0b0000	0b0111	0b001

Accessibility

MRS <Xt>, ID_AA64MMFR1_EL1

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

B.5.36 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-90: AArch64_id_aa64mmfr2_el1 bit assignments

63–60	59–56	55–52	51–48	47–44	43–40	39–36	35–32
E0PD	EVT	BBM	TTL	RES0	FWB	IDS	AT

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
ST	NV	CCIDX	RES0	IESB	RES0	UAO	CnP

Table B-241: 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.	
[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.	
[55:52]	BBM	Allows identification of the requirements of the hardware to have break-before-make sequences when changing block size for a translation. **`0b0010`** Level 2 support for changing block size is supported.	
[51:48]	TTL	Indicates support for TTL field in address operations. Defined values are: **`0b0001`** TLB maintenance instructions by address have bits[47:44] holding the TTL field.	
[47:44]	RES0	Reserved	`0b0000`
[43:40]	FWB	Indicates support for AArch64-HCR_EL2.FWB. Defined values are: **`0b0001`** AArch64-HCR_EL2.FWB is supported.	
[39:36]	IDS	Indicates the value of ESR_ELx.EC that reports an exception generated by a read access to the feature ID space. Defined values are: **`0b0001`** All exceptions generated by an AArch64 read access to the feature ID space are reported by ESR_ELx.EC == 0x18.	

Bits	Name	Description	Reset
[35:32]	AT	Identifies support for unaligned single-copy atomicity and atomic functions. Defined values are: **0b0001** Unaligned single-copy atomicity and atomic functions with a 16-byte address range aligned to 16-bytes are supported.	
[31:28]	ST	Identifies support for small translation tables. Defined values are: **0b0001** The maximum value of the TCR_ELx.{T0SZ,T1SZ} and VTCR_EL2.T0SZ fields is 48 for 4KB and 16KB granules, and 47 for 64KB granules.	
[27:24]	NV	Nested Virtualization. If EL2 is implemented, indicates support for the use of nested virtualization. Defined values are: **0b0010** The AArch64-VNCR_EL2 register and the HCR_EL2.{AT, NV, NV1, NV2} bits are implemented.	
[23:20]	CCIDX	Support for the use of revised AArch64-CCSIDR_EL1 register format. Defined values are: **0b0001** 64-bit format implemented for all levels of the CCSIDR_EL1.	
[19:16]	RES0	Reserved	0b0000
[15:12]	IESB	Indicates support for the IESB bit in the SCTLR_ELx registers. Defined values are: **0b0001** IESB bit in the SCTLR_ELx registers is supported.	
[11:8]	RES0	Reserved	0b0000
[7:4]	UAO	User Access Override. Defined values are: **0b0001** UAO supported.	
[3:0]	CnP	Indicates support for Common not Private translations. Defined values are: **0b0001** Common not Private translations supported.	

Access

MRS <Xt>, ID_AA64MMFR2_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64MMFR2_EL1	0b11	0b000	0b0000	0b0111	0b010

Accessibility

MRS <Xt>, ID_AA64MMFR2_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && (!IsZero(ID_AA64MMFR2_EL1) || boolean IMPLEMENTATION_DEFINED
 "ID_AA64MMFR2 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFR2_EL1;
```

```
elsif PSTATE.EL == EL2 then
    return ID_AA64MMFR2_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64MMFR2_EL1;
```

B.5.37 CLIDR_EL1, Cache Level ID Register

Identifies the type of cache, or caches, that are implemented at each level and can be managed using the architected cache maintenance instructions that operate by set/way, up to a maximum of seven levels. Also identifies the Level of Coherence (LoC) and Level of Unification (LoU) for the cache hierarchy.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-91: AArch64_clidr_el1 bit assignments

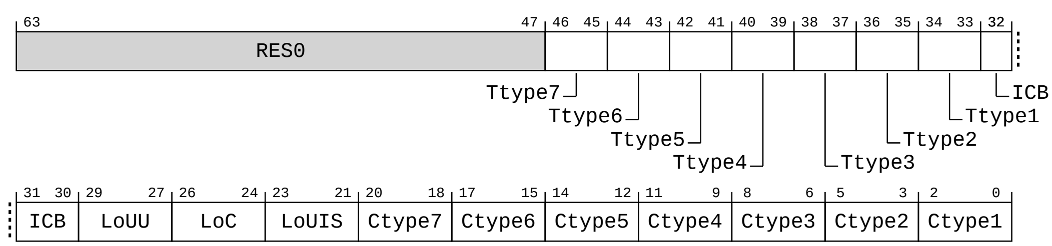


Table B-243: CLIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:47]	RES0	Reserved	`0x0`
[46:45]	Ttype7	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **`0b00`** No Tag Cache.	

Bits	Name	Description	Reset
[44:43]	Ttype6	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **`0b00`** No Tag Cache.	
[42:41]	Ttype5	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **`0b00`** No Tag Cache.	
[40:39]	Ttype4	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **`0b00`** No Tag Cache.	
[38:37]	Ttype3	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **`0b00`** When no L3 present, no tag cache. **`0b10`** When L3 present, Unified Allocation Tag and Data cache at L3	
[36:35]	Ttype2	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **`0b10`** Unified Allocation Tag and Data cache at L1	
[34:33]	Ttype1	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **`0b10`** Unified Allocation Tag and Data cache at L1	

Bits	Name	Description	Reset
[32:30]	ICB	Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions. The possible values are: **`0b000`** Not disclosed by this mechanism. **`0b001`** L1 cache is the highest Inner Cacheable level. **`0b010`** L2 cache is the highest Inner Cacheable level. **`0b011`** L3 cache is the highest Inner Cacheable level. **`0b100`** L4 cache is the highest Inner Cacheable level. **`0b101`** L5 cache is the highest Inner Cacheable level. **`0b110`** L6 cache is the highest Inner Cacheable level. **`0b111`** L7 cache is the highest Inner Cacheable level.	
[29:27]	LoUU	Level of Unification Uniprocessor for the cache hierarchy. **`0b000`** Level of Unification Uniprocessor is before the L1 D-cache.	
[26:24]	LoC	Level of Coherence for the cache hierarchy. **`0b010`** When no L3 present, Level 2 **`0b011`** When L3 present, Level 3	
[23:21]	LoUIS	Level of Unification Inner Shareable for the cache hierarchy. **`0b000`** No cache level needs cleaning to Point of Unification	
[20:18]	Ctype7	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **`0b000`** No cache.	
[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.	

Bits	Name	Description	Reset
[14:12]	Ctype5	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **`0b000`** No cache.	
[11:9]	Ctype4	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **`0b000`** No cache.	
[8:6]	Ctype3	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **`0b000`** No L3. **`0b100`** Unified instruction and data caches at L3	
[5:3]	Ctype2	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **`0b100`** Unified instruction and data caches at L2	
[2:0]	Ctype1	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **`0b011`** Separate instruction and data caches at L1	

Access

MRS <Xt>, CLIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
CLIDR_EL1	`0b11`	`0b001`	`0b0000`	`0b0000`	`0b001`

Accessibility

MRS <Xt>, CLIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.CLIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    else
        return CLIDR_EL1;
elsif PSTATE.EL == EL2 then
    return CLIDR_EL1;
elsif PSTATE.EL == EL3 then
    return CLIDR_EL1;
```

B.5.38 GMID_EL1, Multiple tag transfer ID register

Indicates the block size that is accessed by the LDGM and STGM System instructions.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-92: AArch64_gmid_el1 bit assignments

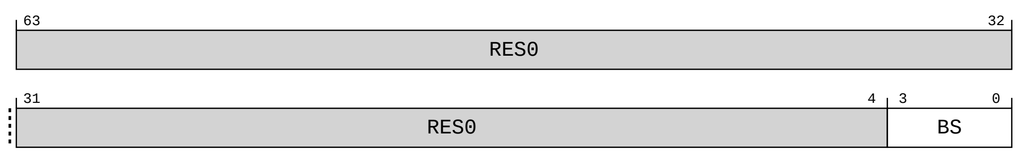


Table B-245: GMID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	`0x0`
[3:0]	BS	Log_2 of the block size in words. The minimum supported size is 16B (value == 2) and the maximum is 256B (value == 6). **`0b0100`** Log2 of the block size is 4	

Access

MRS <Xt>, GMID_EL1

<systemreg>	op0	op1	CRn	CRm	op2
GMID_EL1	`0b11`	`0b001`	`0b0000`	`0b0000`	`0b100`

Accessibility

MRS <Xt>, GMID_EL1

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

B.5.39 CTR_EL0, Cache Type Register

Provides information about the architecture of the caches.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-93: AArch64_ctr_el0 bit assignments

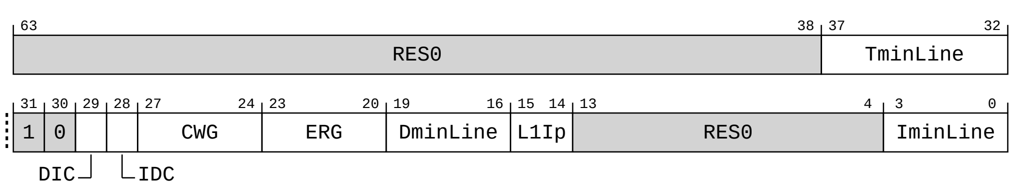


Table B-247: CTR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:38]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[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. **Note:** • For an implementation with cache lines containing 64 bytes of data and 4 Allocation Tags, this will be $log_2(64/4) = 4$. • For an implementation with Allocations Tags in separate cache lines of 128 Allocation Tags per line, this will be $log_2(128*16/4) = 9$. `0b000100` Log2 of number of words (64/4=16) covered by Allocation Tags in the smallest cache line of all caches	
[31]	RES1	Reserved	`0b1`
[30]	RES0	Reserved	`0b0`

[29]	DIC	Instruction cache invalidation requirements for data to instruction coherence.
		
		`0b0`
		When COHERENT_ICACHE not enabled, Instruction cache invalidation to the point of unification is required for instruction to data coherence.
		
		`0b1`
		When COHERENT_ICACHE enabled, Instruction cache cleaning to the point of unification is not required for instruction to data coherence.	

[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.	
[27:24]	CWG	Cache writeback granule. Log2 of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified. `0b0100` 64 bytes.	
[23:20]	ERG	Exclusives reservation granule, and, if TME is implemented, transactional reservation granule. Log2 of the number of words of the maximum size of the reservation granule for the Load-Exclusive and Store-Exclusive instructions, and, if TME is implemented, for detecting transactional conflicts. `0b0100` 64 bytes.	
[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.	
[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)	
[13:4]	RES0	Reserved	`0x0`
[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.	

Access

MRS <Xt>, CTR_EL0

<systemreg>	op0	op1	CRn	CRm	op2
CTR_EL0	0b11	0b011	0b0000	0b0000	0b001

Accessibility

MRS <Xt>, CTR_EL0

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.UCT == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' && HF\
GRTR_EL2.CTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CTR_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.CTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CTR_EL0;
elsif PSTATE.EL == EL2 then
    return CTR_EL0;
elsif PSTATE.EL == EL3 then
    return CTR_EL0;
```

B.5.40 DCZID_EL0, Data Cache Zero ID register

Indicates the block size that is written with byte values of 0 by the rDC ZVA (Data Cache Zero by Address) System instruction.

If ARMv8.5-MemTag is implemented, this register also indicates the granularity at which the rDC GVA and rDC GZVA instructions write.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-94: AArch64_dczid_el0 bit assignments

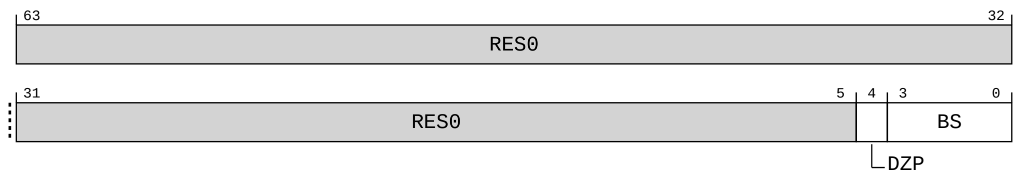


Table B-249: DCZID_EL0 bit descriptions

Bits	Name	Description	Reset
[63:5]	RES0	Reserved	`0x0`
[4]	DZP	Data Zero Prohibited. This field indicates whether use of rDC ZVA instructions is permitted or prohibited. If ARMv8.5-MemTag is implemented, this field also indicates whether use of the rDC GVA and rDC GZVA instructions are permitted or prohibited. **`0b0`** Instructions are permitted.	
[3:0]	BS	Log_2 of the block size in words. The maximum size supported is 2KB (value == 9). **`0b0100`** Log2 of the block size is 4	

Access

MRS <Xt>, DCZID_EL0

<systemreg>	op0	op1	CRn	CRm	op2
DCZID_EL0	`0b11`	`0b011`	`0b0000`	`0b0000`	`0b111`

Accessibility

MRS <Xt>, DCZID_EL0

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' && HF\
GRTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return DCZID_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return DCZID_EL0;
elsif PSTATE.EL == EL2 then
    return DCZID_EL0;
elsif PSTATE.EL == EL3 then
    return DCZID_EL0;
```

B.5.41 MPAMIDR_EL1, MPAM ID Register (EL1)

Indicates the presence and maximum PARTID and PMG values supported in the implementation. It also indicates whether the implementation supports MPAM virtualization.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

MPAMIDR_EL1 indicates the MPAM implementation parameters of the PE.

Figure B-95: AArch64_mpamidr_el1 bit assignments

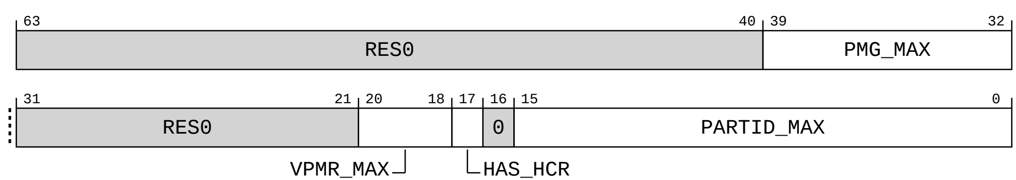


Table B-251: MPAMIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	`0x0`
[39:32]	PMG_MAX	The largest value of PMG that the implementation can generate. The PMG_I and PMG_D fields of every MPAMn_ELx must implement at least enough bits to represent PMG_MAX. **`0b00000001`** Max PMG field is 1 (1-bit)	
[31:21]	RES0	Reserved	`0x0`
[20:18]	VPMR_MAX	If HAS_HCR == 0, VPMR_MAX must be 000. Otherwise, it indicates the maximum register index n for the MPAMVPM<n>_EL2 registers. **`0b111`** 8 MPAMVPMn_EL2 registers are implemented	

Bits	Name	Description	Reset
[17]	HAS_HCR	HAS_HCR indicates that the PE implementation supports MPAM virtualization, including AArch64-MPAMHCR_EL2, AArch64-MPAMVPMV_EL2 and MPAMVPM<n>_EL2 with n in the range 0 to VPMR_MAX. Must be 0 if EL2 is not implemented in either security state. **0b1** MPAM virtualization is supported.	
[16]	RES0	Reserved	0b0
[15:0]	PARTID_MAX	The largest value of PARTID that the implementation can generate. The PARTID_I and PARTID_D fields of every MPAMn_ELx must implement at least enough bits to represent PARTID_MAX. **0b0000000111111111** Max PARTID field is 511	

Access

MRS <Xt>, MPAMIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MPAMIDR_EL1	0b11	0b000	0b1010	0b0100	0b100

Accessibility

MRS <Xt>, MPAMIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        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
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMIDR_EL1;
elsif PSTATE.EL == EL3 then
    return MPAMIDR_EL1;
```

B.5.42 IMP_CPUCFR_EL1, CPU Configuration Register

This register provides configuration information for the core.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-96: AArch64_imp_cpucfr_el1 bit assignments

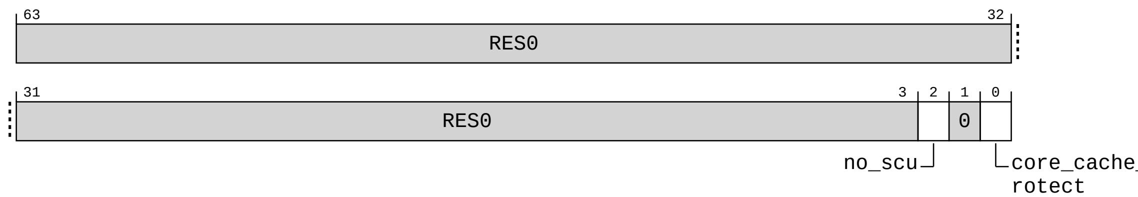


Table B-253: IMP_CPUCFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	0x0
[2]	no_scu	Indicates whether the SCU is present or not. Possible values of this bit are: **0b0** The SCU is present. **0b1** The SCU is not present.	
[1]	RES0	Reserved	0x0
[0]	core_cache_protect	Indicates whether ECC is present or not. Possible values of this field are: **0b0** ECC is not present. **0b1** ECC is present.	

Access

MRS <Xt>, S3_0_C15_C0_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C0_0	0b11	0b000	0b1111	0b0000	0b000

Accessibility

MRS <Xt>, S3_0_C15_C0_0

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

```
        return IMP_CPUCFR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUCFR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUCFR_EL1;
```

B.6 AArch64 performance monitors registers

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

Table B-255: performance-monitors register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
PMMIR_EL1	3	C9	0	C14	6	See individual bit resets.	64-bit	Performance Monitors Machine Identification Register
PMCR_EL0	3	C9	3	C12	0	See individual bit resets.	64-bit	Performance Monitors Control Register
PMCEID0_EL0	3	C9	3	C12	6	See individual bit resets.	64-bit	Performance Monitors Common Event Identification register 0
PMCEID1_EL0	3	C9	3	C12	7	See individual bit resets.	64-bit	Performance Monitors Common Event Identification register 1

B.6.1 PMMIR_EL1, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation to software.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

performance-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-97: AArch64_pmmir_el1 bit assignments

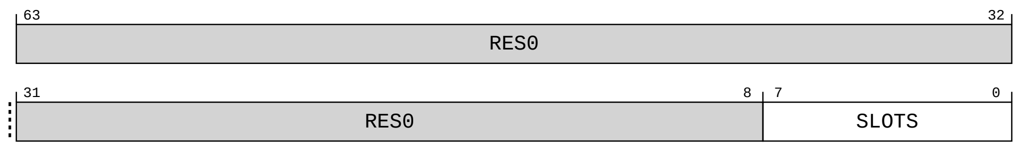


Table B-256: PMMIR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	0x0
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero. **0b00000101** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 5.	

Access

MRS <Xt>, PMMIR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
PMMIR_EL1	0b11	0b000	0b1001	0b1110	0b110

Accessibility

MRS <Xt>, PMMIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMMIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMMIR_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMMIR_EL1;
elsif PSTATE.EL == EL3 then
    return PMMIR_EL1;
```

B.6.2 PMCR_EL0, Performance Monitors Control Register

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

performance-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-98: AArch64_pmcr_el0 bit assignments

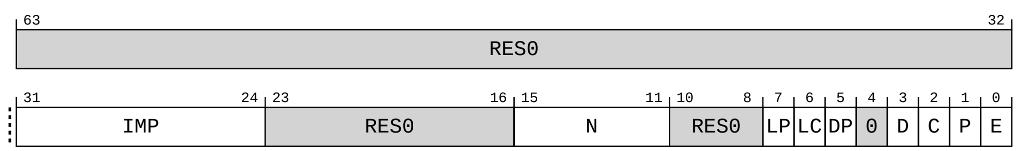


Table B-258: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:24]	IMP	Implementer code: **`0b00000000`** No ID information is present in PMCR/PMCR_EL0. Software must use the MIDR_EL1 to identify the PE.	
[23:16]	RES0	Reserved	`0b00000000`
[15:11]	N	Number of event counters: **`0b00110`** 6 PMU counters implemented	
[10:8]	RES0	Reserved	`0b000`

Bits	Name	Description	Reset
[7]	LP	Long event counter enable. Determines when unsigned overflow is recorded by a 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.	
[6]	LC	Long cycle counter enable. Determines when unsigned overflow is recorded by the cycle counter overflow bit. **0b0** Cycle counter overflow on increment that causes unsigned overflow of AArch64-PMCCNTR_EL0[31:0]. **0b1** Cycle counter overflow on increment that causes unsigned overflow of AArch64-PMCCNTR_EL0[63:0]. Arm deprecates use of AArch64-PMCR_EL0.LC = 0.	
[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.	
[4]	RES0	Reserved	0b0
[3]	D	Clock divider. **0b0** When enabled, AArch64-PMCCNTR_EL0 counts every clock cycle. **0b1** When enabled, AArch64-PMCCNTR_EL0 counts once every 64 clock cycles. If PMCR_EL0.LC == 1, this bit is ignored and the cycle counter counts every clock cycle. Arm deprecates use of PMCR_EL0.D = 1.	

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. This bit is always RAZ. **Note:** Resetting AArch64-PMCCNTR_EL0 does not change the cycle counter overflow bit. The value of PMCR_EL0.LC is ignored, and bits [63:0] of all affected event counters are reset.	
[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. This bit is always RAZ. 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 ARMv8.5-PMU 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.	

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 AArch32, PMN is AArch32-HDCR.HPMN. • 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.	

Access

MRS <Xt>, PMCR_EL0

<systemreg>	op0	op1	CRn	CRm	op2
PMCR_EL0	0b11	0b011	0b1001	0b1100	0b000

MSR PMCR_EL0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
PMCR_EL0	0b11	0b011	0b1001	0b1100	0b000

Accessibility

MRS <Xt>, PMCR_EL0

```
if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCR_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCR_EL0;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCR_EL0;
elsif PSTATE.EL == EL3 then
    return PMCR_EL0;
```

MSR PMCR_EL0, <Xt>

```
if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' && HD\
FGWTR_EL2.PMCR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMCR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t];
elsif PSTATE.EL == EL3 then
    PMCR_EL0 = X[t];
```

B.6.3 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.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. For more information about the common events and the use of the PMCEID<n>_EL0 registers see 'The PMU event number space and common events'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

performance-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-99: AArch64_pmceid0_el0 bit assignments

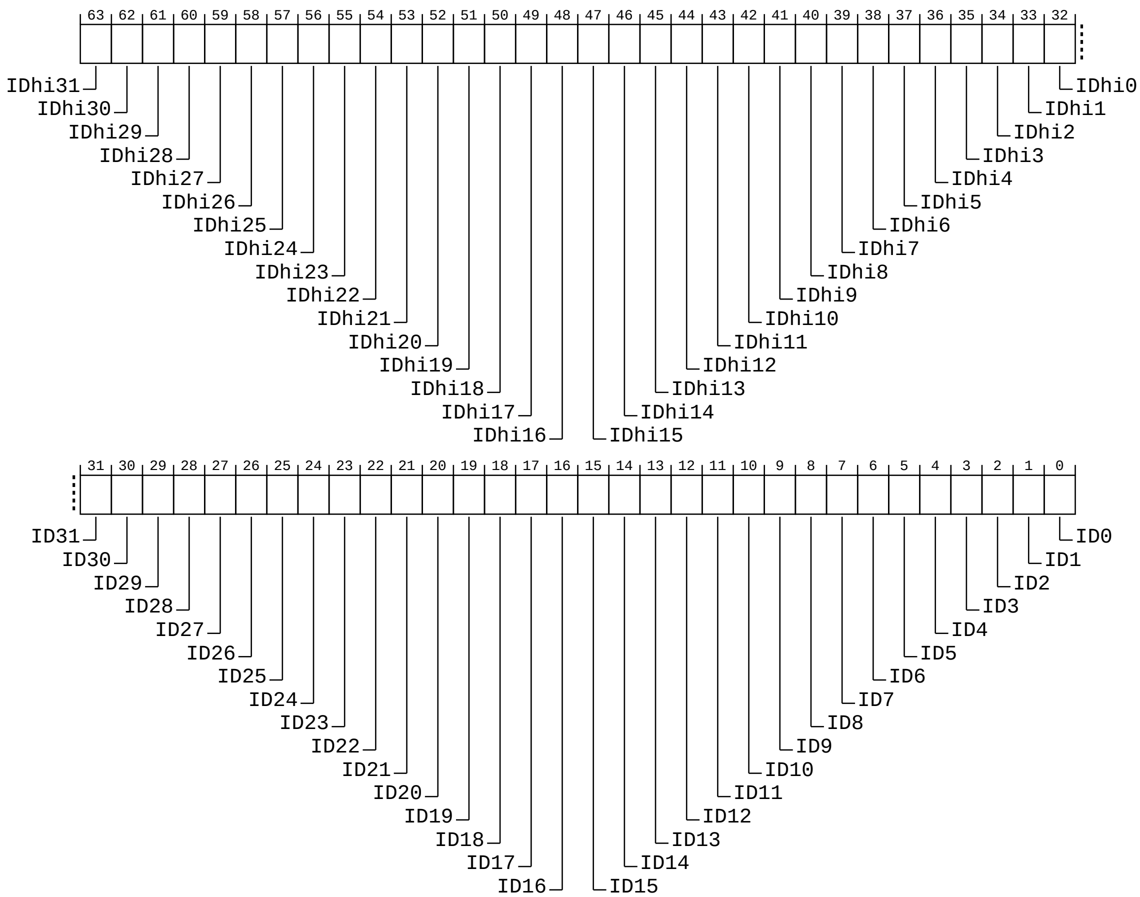


Table B-261: PMCEID0_EL0 bit descriptions

Bits	Name	Description	Reset
[63]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x401f) **0b0** The common event is not implemented, or not counted.	
[62]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x401e) **0b0** The common event is not implemented, or not counted.	
[61]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x401d) **0b0** The common event is not implemented, or not counted.	
[60]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x401c) **0b0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[59]	IDhi27	IDhi27 corresponds to common event (0x401b) CTI_TRIGOUT7 **0b1** The common event is implemented.	
[58]	IDhi26	IDhi26 corresponds to common event (0x401a) CTI_TRIGOUT6 **0b1** The common event is implemented.	
[57]	IDhi25	IDhi25 corresponds to common event (`0x4019`) CTI_TRIGOUT5 **0b1** The common event is implemented.	
[56]	IDhi24	IDhi24 corresponds to common event (`0x4018`) CTI_TRIGOUT4 **0b1** The common event is implemented.	
[55]	IDhi23	IDhi23 corresponds to a Reserved Event event (`0x4017`) **0b0** The common event is not implemented, or not counted.	
[54]	IDhi22	IDhi22 corresponds to a Reserved Event event (`0x4016`) **0b0** The common event is not implemented, or not counted.	
[53]	IDhi21	IDhi21 corresponds to a Reserved Event event (`0x4015`) **0b0** The common event is not implemented, or not counted.	
[52]	IDhi20	IDhi20 corresponds to a Reserved Event event (`0x4014`) **0b0** The common event is not implemented, or not counted.	
[51]	IDhi19	IDhi19 corresponds to common event (`0x4013`) TRCEXTOUT3 **0b1** The common event is implemented.	
[50]	IDhi18	IDhi18 corresponds to common event (`0x4012`) TRCEXTOUT2 **0b1** The common event is implemented.	
[49]	IDhi17	IDhi17 corresponds to common event (`0x4011`) TRCEXTOUT1 **0b1** The common event is implemented.	
[48]	IDhi16	IDhi16 corresponds to common event (`0x4010`) TRCEXTOUT0 **0b1** The common event is implemented.	
[47]	IDhi15	IDhi15 corresponds to common event (0x400f) Reserved **0b0** The common event is not implemented, or not counted.	
[46]	IDhi14	IDhi14 corresponds to common event (0x400e) TRB_TRIG **0b0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[45]	IDhi13	IDhi13 corresponds to common event (0x400d) PMU_OVFS **0b0** The common event is not implemented, or not counted.	
[44]	IDhi12	IDhi12 corresponds to common event (0x400c) TRB_WRAP **0b1** The common event is implemented.	
[43]	IDhi11	IDhi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD **0b1** The common event is implemented.	
[42]	IDhi10	IDhi10 corresponds to common event (0x400a) L2I_CACHE_LMISS **0b0** The common event is not implemented, or not counted.	
[41]	IDhi9	IDhi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD **0b1** The common event is implemented.	
[40]	IDhi8	IDhi8 corresponds to common event (0x4008) Reserved **0b0** The common event is not implemented, or not counted.	
[39]	IDhi7	IDhi7 corresponds to common event (0x4007) Reserved **0b0** The common event is not implemented, or not counted.	
[38]	IDhi6	IDhi6 corresponds to common event (0x4006) L1I_CACHE_LMISS **0b1** The common event is implemented.	
[37]	IDhi5	IDhi5 corresponds to common event (0x4005) STALL_BACKEND_MEM **0b1** The common event is implemented.	
[36]	IDhi4	IDhi4 corresponds to common event (0x4004) CNT_CYCLES **0b1** The common event is implemented.	
[35]	IDhi3	IDhi3 corresponds to common event (0x4003) SAMPLE_COLLISION **0b1** The common event is implemented.	
[34]	IDhi2	IDhi2 corresponds to common event (0x4002) SAMPLE_FILTRATE **0b1** The common event is implemented.	
[33]	IDhi1	IDhi1 corresponds to common event (0x4001) SAMPLE_FEED **0b1** The common event is implemented.	
[32]	IDhi0	IDhi0 corresponds to common event (0x4000) SAMPLE_POP **0b1** The common event is implemented.	

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x1f) L1D_CACHE_ALLOCATE **0b0** The common event is not implemented, or not counted.	
[30]	ID30	ID30 corresponds to common event (0x1e) CHAIN **0b1** The common event is implemented.	
[29]	ID29	ID29 corresponds to common event (0x1d) BUS_CYCLES **0b1** The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED **0b1** The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x1b) INST_SPEC **0b1** The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x1a) MEMORY_ERROR **0b1** The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (0x19) BUS_ACCESS **0b1** The common event is implemented.	
[24]	ID24	ID24 corresponds to common event (0x18) L2D_CACHE_WB **0b1** The common event is implemented.	
[23]	ID23	ID23 corresponds to common event (0x17) L2D_CACHE_REFILL **0b1** The common event is implemented.	
[22]	ID22	ID22 corresponds to common event (0x16) L2D_CACHE **0b1** The common event is implemented.	
[21]	ID21	ID21 corresponds to common event (0x15) L1D_CACHE_WB **0b1** The common event is implemented.	
[20]	ID20	ID20 corresponds to common event (0x14) L1I_CACHE **0b1** The common event is implemented.	
[19]	ID19	ID19 corresponds to common event (0x13) MEM_ACCESS **0b1** The common event is implemented.	
[18]	ID18	ID18 corresponds to common event (0x12) BR_PRED **0b1** The common event is implemented.	

Bits	Name	Description	Reset
[17]	ID17	ID17 corresponds to common event (`0x11`) CPU_CYCLES **`0b1`** The common event is implemented.	
[16]	ID16	ID16 corresponds to common event (`0x10`) BR_MIS_PRED **`0b1`** The common event is implemented.	
[15]	ID15	ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[14]	ID14	ID14 corresponds to common event (0xe) BR_RETURN_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[12]	ID12	ID12 corresponds to common event (0xc) PC_WRITE_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[11]	ID11	ID11 corresponds to common event (0xb) CID_WRITE_RETIRED **`0b1`** The common event is implemented.	
[10]	ID10	ID10 corresponds to common event (0xa) EXC_RETURN **`0b1`** The common event is implemented.	
[9]	ID9	ID9 corresponds to common event (`0x9`) EXC_TAKEN **`0b1`** The common event is implemented.	
[8]	ID8	ID8 corresponds to common event (`0x8`) INST_RETIRED **`0b1`** The common event is implemented.	
[7]	ID7	ID7 corresponds to common event (`0x7`) ST_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[6]	ID6	ID6 corresponds to common event (`0x6`) LD_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[5]	ID5	ID5 corresponds to common event (`0x5`) L1D_TLB_REFILL **`0b1`** The common event is implemented.	
[4]	ID4	ID4 corresponds to common event (`0x4`) L1D_CACHE **`0b1`** The common event is implemented.	

Bits	Name	Description	Reset
[3]	ID3	ID3 corresponds to common event (`0x3`) L1D_CACHE_REFILL **`0b1`** The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (`0x2`) L1I_TLB_REFILL **`0b1`** The common event is implemented.	
[1]	ID1	ID1 corresponds to common event (`0x1`) L1I_CACHE_REFILL **`0b1`** The common event is implemented.	
[0]	ID0	ID0 corresponds to common event (`0x0`) SW_INCR **`0b1`** The common event is implemented.	

Access

MRS <Xt>, PMCEID0_EL0

<systemreg>	op0	op1	CRn	CRm	op2
PMCEID0_EL0	`0b11`	`0b011`	`0b1001`	`0b1100`	`0b110`

Accessibility

MRS <Xt>, PMCEID0_EL0

```
if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID0_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID0_EL0;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID0_EL0;
elsif PSTATE.EL == EL3 then
    return PMCEID0_EL0;
```

B.6.4 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.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. For more information about the common events and the use of the PMCEID<n>_EL0 registers see 'The PMU event number space and common events'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

performance-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-100: AArch64_pmceid1_el0 bit assignments

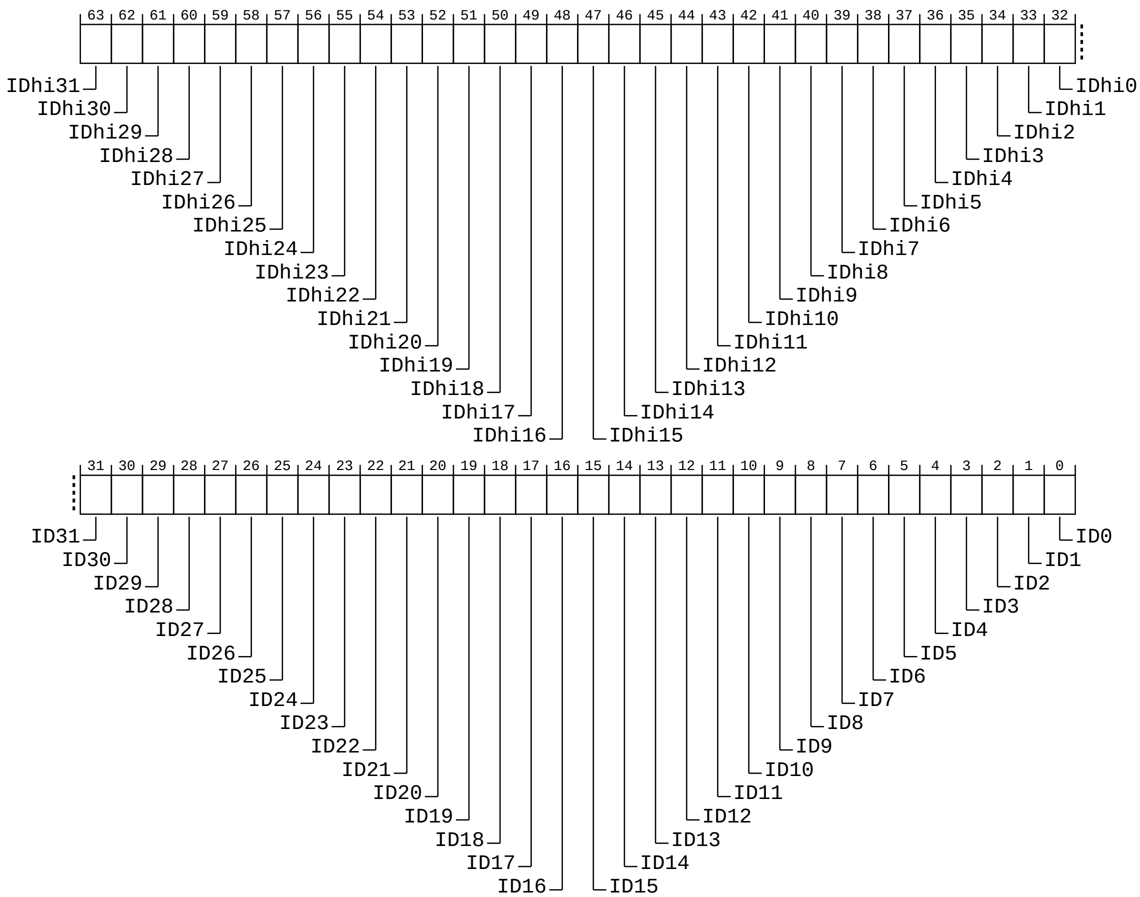


Table B-263: PMCEID1_EL0 bit descriptions

Bits	Name	Description	Reset
[63]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x403f) **0b0** The common event is not implemented, or not counted.	
[62]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x403e) **0b0** The common event is not implemented, or not counted.	
[61]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x403d) **0b0** The common event is not implemented, or not counted.	
[60]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x403c) **0b0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[59]	IDhi27	IDhi27 corresponds to a Reserved Event event (0x403b) **0b0** The common event is not implemented, or not counted.	
[58]	IDhi26	IDhi26 corresponds to a Reserved Event event (0x403a) **0b0** The common event is not implemented, or not counted.	
[57]	IDhi25	IDhi25 corresponds to a Reserved Event event (0x4039) **0b0** The common event is not implemented, or not counted.	
[56]	IDhi24	IDhi24 corresponds to a Reserved Event event (0x4038) **0b0** The common event is not implemented, or not counted.	
[55]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4037) **0b0** The common event is not implemented, or not counted.	
[54]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4036) **0b0** The common event is not implemented, or not counted.	
[53]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4035) **0b0** The common event is not implemented, or not counted.	
[52]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4034) **0b0** The common event is not implemented, or not counted.	
[51]	IDhi19	IDhi19 corresponds to a Reserved Event event (0x4033) **0b0** The common event is not implemented, or not counted.	
[50]	IDhi18	IDhi18 corresponds to a Reserved Event event (0x4032) **0b0** The common event is not implemented, or not counted.	
[49]	IDhi17	IDhi17 corresponds to a Reserved Event event (0x4031) **0b0** The common event is not implemented, or not counted.	
[48]	IDhi16	IDhi16 corresponds to a Reserved Event event (0x4030) **0b0** The common event is not implemented, or not counted.	
[47]	IDhi15	IDhi15 corresponds to a Reserved Event event (0x402f) **0b0** The common event is not implemented, or not counted.	
[46]	IDhi14	IDhi14 corresponds to a Reserved Event event (0x402e) **0b0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[45]	IDhi13	IDhi13 corresponds to a Reserved Event event (0x402d) **0b0** The common event is not implemented, or not counted.	
[44]	IDhi12	IDhi12 corresponds to a Reserved Event event (0x402c) **0b0** The common event is not implemented, or not counted.	
[43]	IDhi11	IDhi11 corresponds to a Reserved Event event (0x402b) **0b0** The common event is not implemented, or not counted.	
[42]	IDhi10	IDhi10 corresponds to a Reserved Event event (0x402a) **0b0** The common event is not implemented, or not counted.	
[41]	IDhi9	IDhi9 corresponds to a Reserved Event event (0x4029) **0b0** The common event is not implemented, or not counted.	
[40]	IDhi8	IDhi8 corresponds to a Reserved Event event (0x4028) **0b0** The common event is not implemented, or not counted.	
[39]	IDhi7	IDhi7 corresponds to a Reserved Event event (0x4027) **0b0** The common event is not implemented, or not counted.	
[38]	IDhi6	IDhi6 corresponds to common event (0x4026) MEM_ACCESS_CHECKED_WR **0b1** The common event is implemented.	
[37]	IDhi5	IDhi5 corresponds to common event (0x4025) MEM_ACCESS_CHECKED_RD **0b1** The common event is implemented.	
[36]	IDhi4	IDhi4 corresponds to common event (0x4024) MEM_ACCESS_CHECKED **0b1** The common event is implemented.	
[35]	IDhi3	IDhi3 corresponds to common event (0x4023) Reserved **0b0** The common event is not implemented, or not counted.	
[34]	IDhi2	IDhi2 corresponds to common event (0x4022) ST_ALIGN_LAT **0b1** The common event is implemented.	
[33]	IDhi1	IDhi1 corresponds to common event (0x4021) LD_ALIGN_LAT **0b1** The common event is implemented.	
[32]	IDhi0	IDhi0 corresponds to common event (0x4020) LDST_ALIGN_LAT **0b1** The common event is implemented.	

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x3f) STALL_SLOT **0b1** The common event is implemented.	
[30]	ID30	ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND **0b1** The common event is implemented.	
[29]	ID29	ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND **0b1** The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x3c) STALL **0b1** The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x3b) OP_SPEC **0b1** The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x3a) OP_RETIRED **0b1** The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (0x39) L1D_CACHE_LMISS_RD **0b1** The common event is implemented.	
[24]	ID24	ID24 corresponds to common event (0x38) REMOTE_ACCESS_RD **0b0** The common event is not implemented, or not counted.	
[23]	ID23	ID23 corresponds to common event (0x37) LL_CACHE_MISS_RD **0b1** The common event is implemented.	
[22]	ID22	ID22 corresponds to common event (0x36) LL_CACHE_RD **0b1** The common event is implemented.	
[21]	ID21	ID21 corresponds to common event (0x35) ITLB_WALK **0b1** The common event is implemented.	
[20]	ID20	ID20 corresponds to common event (0x34) DTLB_WALK **0b1** The common event is implemented.	
[19]	ID19	ID19 corresponds to a Reserved Event event (0x33) **0b0** The common event is not implemented, or not counted.	
[18]	ID18	ID18 corresponds to a Reserved Event event (0x32) **0b0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[17]	ID17	ID17 corresponds to common event (`0x31`) REMOTE_ACCESS **`0b1`** The common event is implemented.	
[16]	ID16	ID16 corresponds to common event (`0x30`) L2I_TLB **`0b0`** The common event is not implemented, or not counted.	
[15]	ID15	ID15 corresponds to common event (0x2f) L2D_TLB **`0b1`** The common event is implemented.	
[14]	ID14	ID14 corresponds to common event (0x2e) L2I_TLB_REFILL **`0b0`** The common event is not implemented, or not counted.	
[13]	ID13	ID13 corresponds to common event (0x2d) L2D_TLB_REFILL **`0b1`** The common event is implemented.	
[12]	ID12	ID12 corresponds to common event (0x2c) Reserved **`0b0`** The common event is not implemented, or not counted.	
[11]	ID11	ID11 corresponds to common event (0x2b) L3D_CACHE **`0b1`** The common event is implemented.	
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL **`0b1`** The common event is implemented.	
[9]	ID9	ID9 corresponds to common event (`0x29`) L3D_CACHE_ALLOCATE **`0b1`** The common event is implemented.	
[8]	ID8	ID8 corresponds to common event (`0x28`) L2I_CACHE_REFILL **`0b0`** The common event is not implemented, or not counted.	
[7]	ID7	ID7 corresponds to common event (`0x27`) L2I_CACHE **`0b0`** The common event is not implemented, or not counted.	
[6]	ID6	ID6 corresponds to common event (`0x26`) L1I_TLB **`0b1`** The common event is implemented.	
[5]	ID5	ID5 corresponds to common event (`0x25`) L1D_TLB **`0b1`** The common event is implemented.	
[4]	ID4	ID4 corresponds to common event (`0x24`) STALL_BACKEND **`0b1`** The common event is implemented.	

Bits	Name	Description	Reset
[3]	ID3	ID3 corresponds to common event (`0x23`) STALL_FRONTEND **`0b1`** The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (`0x22`) BR_MIS_PRED_RETIRED **`0b1`** The common event is implemented.	
[1]	ID1	ID1 corresponds to common event (`0x21`) BR_RETIRED **`0b1`** The common event is implemented.	
[0]	ID0	ID0 corresponds to common event (`0x20`) L2D_CACHE_ALLOCATE **`0b1`** The common event is implemented.	

Access

MRS <Xt>, PMCEID1_EL0

<systemreg>	op0	op1	CRn	CRm	op2
PMCEID1_EL0	0b11	0b011	0b1001	0b1100	0b111

Accessibility

MRS <Xt>, PMCEID1_EL0

```
if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID1_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID1_EL0;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID1_EL0;
elsif PSTATE.EL == EL3 then
    return PMCEID1_EL0;
```

B.7 AArch64 GIC registers

The summary table provides an overview of all implementation defined GIC registers in the core. Individual register descriptions provide detailed information.

Table B-265: GIC register summary

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

B.7.1 ICC_CTLR_EL1, Interrupt Controller Control Register (EL1)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

gic

Reset value

See individual bit resets.

Bit descriptions

Figure B-101: AArch64_icc_ctlr_el1 bit assignments

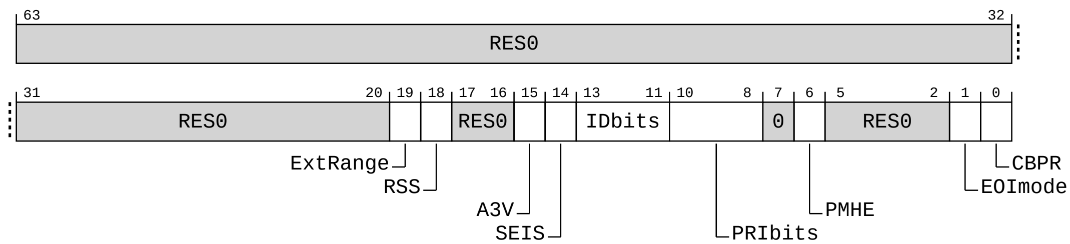


Table B-266: ICC_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	`0x0`
[19]	ExtRange	Extended INTID range (read-only). **`0b1`** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	
[18]	RSS	Range Selector Support. Possible values are: **`0b0`** Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	
[17:16]	RES0	Reserved	`0b00`
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. Possible values are: **`0b1`** The CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.	
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs: **`0b0`** The CPU interface logic does not support local generation of SEIs.	
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported: **`0b000`** 16 bits.	

Bits	Name	Description	Reset
[10:8]	PRIbits	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	
[7]	RES0	Reserved	`0b0`
[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.	
[5:2]	RES0	Reserved	`0b0000`
[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	

Bits	Name	Description	Reset
[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.	

Access

MRS <Xt>, ICC_CTLR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL1	0b11	0b000	0b1100	0b1100	0b100

MSR ICC_CTLR_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL1	0b11	0b000	0b1100	0b1100	0b100

Accessibility

MRS <Xt>, ICC_CTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        return ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        return ICV_CTLR_EL1;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
```

```
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        return ICC_CTLR_EL1_S;
    else
        return ICC_CTLR_EL1_NS;
```

MSR ICC_CTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        ICC_CTLR_EL1_S = X[t];
    else
        ICC_CTLR_EL1_NS = X[t];
```

B.7.2 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register

Controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

gic

Reset value

See individual bit resets.

Bit descriptions

Figure B-102: AArch64_icv_ctlr_el1 bit assignments

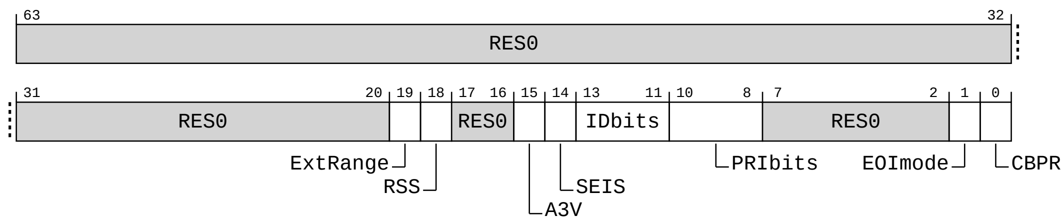


Table B-269: ICV_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	`0x0`
[19]	ExtRange	Extended INTID range (read-only). **`0b1`** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	
[18]	RSS	Range Selector Support. Possible values are: **`0b0`** Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	
[17:16]	RES0	Reserved	`0b00`
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. Possible values are: **`0b1`** The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.	
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the virtual CPU interface supports local generation of SEIs: **`0b0`** The virtual CPU interface logic does not support local generation of SEIs.	
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of virtual interrupt identifier bits supported: **`0b000`** 16 bits.	
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation must implement at least 32 levels of physical priority (5 priority bits). **Note:** This field always returns the number of priority bits implemented. The division between group priority and subpriority is defined in the binary point registers AArch64-ICV_BPR0_EL1 and AArch64-ICV_BPR1_EL1. **`0b100`** 5 bits of priority are implemented	
[7:2]	RES0	Reserved	`0b000000`

Bits	Name	Description	Reset
[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.	
[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** Reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1 plus one, saturated to 0b111. Writes to AArch64-ICV_BPR1_EL1 are ignored.	

Access

MRS <Xt>, ICC_CTLR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL1	0b11	0b000	0b1100	0b1100	0b100

MSR ICC_CTLR_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL1	0b11	0b000	0b1100	0b1100	0b100

Accessibility

MRS <Xt>, ICC_CTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        return ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        return ICV_CTLR_EL1;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
```

```
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        return ICC_CTLR_EL1_S;
    else
        return ICC_CTLR_EL1_NS;
```

MSR ICC_CTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        ICC_CTLR_EL1_S = X[t];
    else
        ICC_CTLR_EL1_NS = X[t];
```

B.7.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Registers

Provides information about Group 0 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

gic

Reset value

See individual bit resets.

Bit descriptions

Figure B-103: AArch64_icc_ap0r0_el1 bit assignments

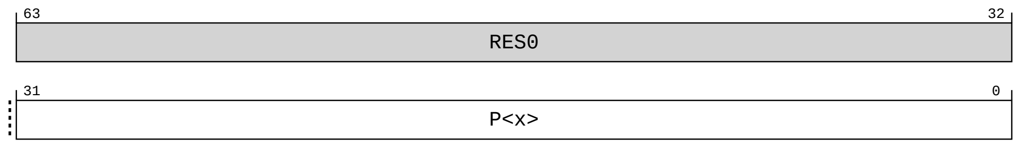


Table B-272: ICC_AP0R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:0]	P<x>	Provides the access to the active priorities for Group 0 interrupts. Possible values of each bit are: **`0b0`** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **`0b1`** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

MRS <Xt>, ICC_AP0R0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP0R0_EL1	`0b11`	`0b000`	`0b1100`	`0b1000`	`0b100`

MSR ICC_AP0R0_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP0R0_EL1	`0b11`	`0b000`	`0b1100`	`0b1000`	`0b100`

B.7.4 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers

Provides information about virtual Group 0 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

gic

Reset value

See individual bit resets.

Bit descriptions

Figure B-104: AArch64_icv_ap0r0_el1 bit assignments

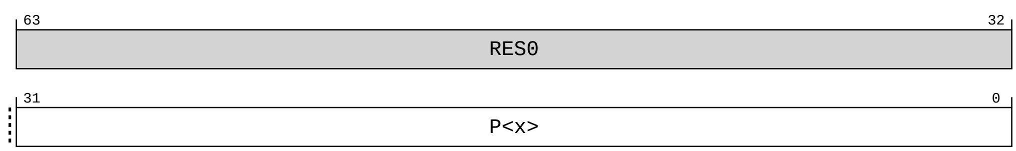


Table B-275: ICV_AP0R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:0]	P<x>	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **`0b0`** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **`0b1`** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

MRS <Xt>, ICC_AP0R0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP0R0_EL1	0b11	0b000	0b1100	0b1000	0b100

MSR ICC_AP0R0_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP0R0_EL1	0b11	0b000	0b1100	0b1000	0b100

B.7.5 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Registers

Provides information about Group 1 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

gic

Reset value

See individual bit resets.

Bit descriptions

Figure B-105: AArch64_icc_ap1r0_el1 bit assignments

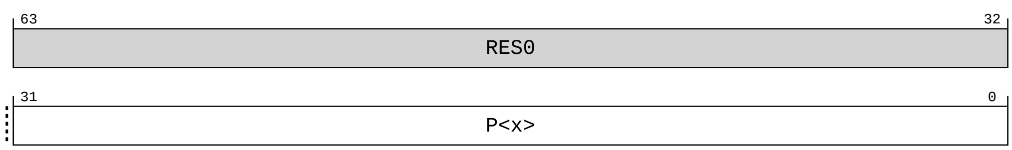


Table B-278: ICC_AP1R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[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.	

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

MRS <Xt>, ICC_AP1R0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP1R0_EL1	`0b11`	`0b000`	`0b1100`	`0b1001`	`0b000`

MSR ICC_AP1R0_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP1R0_EL1	`0b11`	`0b000`	`0b1100`	`0b1001`	`0b000`

B.7.6 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers

Provides information about virtual Group 1 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

gic

Reset value

See individual bit resets.

Bit descriptions

Figure B-106: AArch64_icv_ap1r0_el1 bit assignments

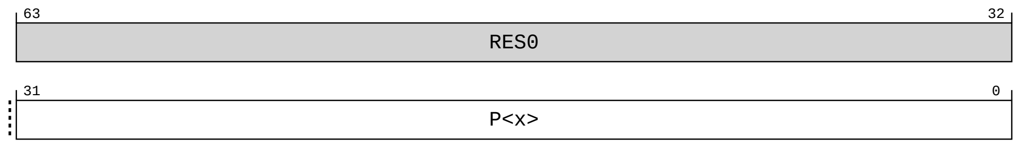


Table B-281: ICV_AP1R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[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].	

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

MRS <Xt>, ICC_AP1R0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP1R0_EL1	`0b11`	`0b000`	`0b1100`	`0b1001`	`0b000`

MSR ICC_AP1R0_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP1R0_EL1	`0b11`	`0b000`	`0b1100`	`0b1001`	`0b000`

B.7.7 ICH_VTR_EL2, Interrupt Controller VGIC Type Register

Reports supported GIC virtualisartion features.

Configurations

If EL2 is not implemented, all bits in this register are RES0 from EL3, except for nV4, which is RES1 from EL3.

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

Attributes

Width

64

Functional group

gic

Reset value

See individual bit resets.

Bit descriptions

Figure B-107: AArch64_ich_vtr_el2 bit assignments

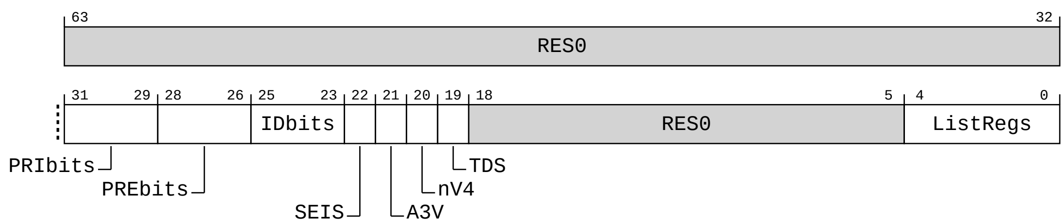


Table B-284: ICH_VTR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[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	
[28:26]	PREbits	The number of virtual preemption bits implemented, minus one. An implementation must implement at least 32 levels of virtual preemption priority (5 preemption bits). The value of this field must be less than or equal to the value of ICH_VTR_EL2.PRIbits. The maximum value of this field is 6, indicating 7 bits of preemption. This field determines the minimum value of AArch64-ICH_VMCR_EL2.VBPR0. **`0b100`** 5 virtual pre-emption bits are implemented	
[25:23]	IDbits	The number of virtual interrupt identifier bits supported: **`0b000`** 16 bits.	

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.	
[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.	
[20]	nV4	Direct injection of virtual interrupts not supported. Possible values are: **0b0** The CPU interface logic supports direct injection of virtual interrupts.	
[19]	TDS	Separate trapping of EL1 writes to AArch64-ICV_DIR_EL1 supported. **0b1** Implementation supports AArch64-ICH_HCR_EL2.TDIR.	
[18:5]	RES0	Reserved	`0x0`
[4:0]	ListRegs	The number of implemented List registers, minus one. For example, a value of 01111 indicates that the maximum of 16 List registers are implemented. **0b00011** 4 List registers	

Access

MRS <Xt>, ICH_VTR_EL2

<systemreg>	op0	op1	CRn	CRm	op2
ICH_VTR_EL2	0b11	0b100	0b1100	0b1011	0b001

Accessibility

MRS <Xt>, ICH_VTR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return ICH_VTR_EL2;
elsif PSTATE.EL == EL3 then
    return ICH_VTR_EL2;
```

B.7.8 ICC_CTLR_EL3, Interrupt Controller Control Register (EL3)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

gic

Reset value

See individual bit resets.

Bit descriptions

Figure B-108: AArch64_icc_ctlr_el3 bit assignments

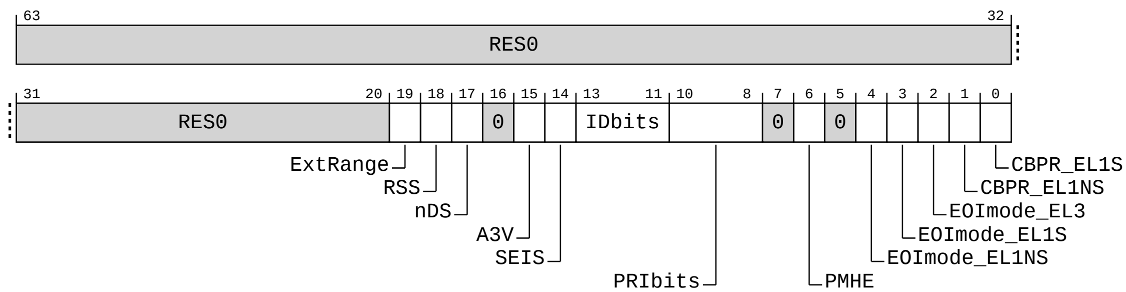


Table B-286: ICC_CTLR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	`0x0`
[19]	ExtRange	Extended INTID range (read-only). **`0b1`** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	
[18]	RSS	Range Selector Support. **`0b0`** Targeted SGIs with affinity level 0 values of 0-15 are supported.	
[17]	nDS	Disable Security not supported. Read-only and writes are ignored. **`0b1`** The CPU interface logic does not support disabling of security, and requires that security is not disabled.	
[16]	RES0	Reserved	`0x0`
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. **`0b1`** The CPU interface logic supports non-zero values of the Aff3 field in SGI generation System registers.	
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports generation of SEIs: **`0b0`** The CPU interface logic does not support generation of SEIs.	

Bits	Name	Description	Reset
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. Indicates the number of physical interrupt identifier bits supported. **0b000** 16 bits.	
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation 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	
[7]	RES0	Reserved	0b0
[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	0b0
[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.	

Bits	Name	Description	Reset
[3]	EOImode_EL1S	EOI mode for interrupts handled at Secure EL1. 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.	
[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.	
[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.	
[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. **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.	

Access

MRS <Xt>, ICC_CTLR_EL3

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL3	0b11	0b110	0b1100	0b1100	0b100

MSR ICC_CTLR_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL3	0b11	0b110	0b1100	0b1100	0b100

Accessibility

MRS <Xt>, ICC_CTLR_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
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];
```

B.8 AArch64 activity monitors registers

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

Table B-289: activity monitors register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
AMEVTYPER10_EL0	3	C13	3	C14	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	C13	3	C14	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	C13	3	C14	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMCFGR_EL0	3	C13	3	C2	1	See individual bit resets.	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	3	C13	3	C2	2	See individual bit resets.	64-bit	Activity Monitors Counter Group Configuration Register

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
AMEVTYPER00_EL0	3	C13	3	C6	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	C13	3	C6	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	C13	3	C6	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	C13	3	C6	3	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0

B.8.1 AMEVTYPER10_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR10_EL0 counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

activity-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-109: AArch64_amevtyper10_el0 bit assignments

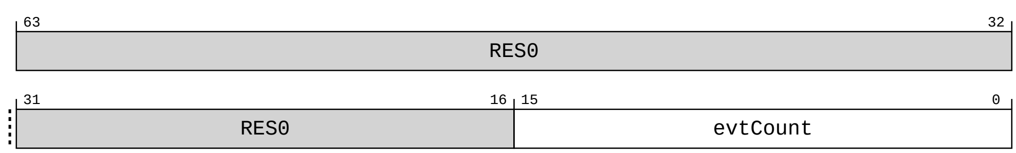


Table B-290: AMEVTYPER10_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	`0x0`

Bits	Name	Description	Reset
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AArch64-AMEVCNTR1<n>_EL0. It is **IMPLEMENTATION DEFINED** what values are supported by each counter. If software writes a value to this field which is not supported by the corresponding counter AArch64-AMEVCNTR1<n>_EL0, then: • It is UNPREDICTABLE which event will be counted. • The value read back is UNKNOWN. The event counted by AArch64-AMEVCNTR1<n>_EL0 might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED. If the corresponding counter AArch64-AMEVCNTR1<n>_EL0 is enabled, writes to this register have UNPREDICTABLE results.	

Access

MRS <Xt>, AMEVTYPER10_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER10_EL0	0b11	0b011	0b1101	0b1110	0b000

MSR AMEVTYPER10_EL0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER10_EL0	0b11	0b011	0b1101	0b1110	0b000

B.8.2 AMEVTYPER11_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR11_EL0 counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

activity-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-110: AArch64_amevtyper11_el0 bit assignments

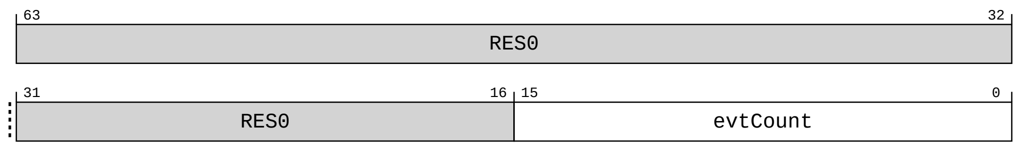


Table B-293: AMEVTYPER11_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	`0x0`
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AArch64-AMEVCNTR1<n>_EL0. It is **IMPLEMENTATION DEFINED** what values are supported by each counter. If software writes a value to this field which is not supported by the corresponding counter AArch64-AMEVCNTR1<n>_EL0, then: • It is UNPREDICTABLE which event will be counted. • The value read back is UNKNOWN. The event counted by AArch64-AMEVCNTR1<n>_EL0 might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED. If the corresponding counter AArch64-AMEVCNTR1<n>_EL0 is enabled, writes to this register have UNPREDICTABLE results.	

Access

MRS <Xt>, AMEVTYPER11_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER11_EL0	`0b11`	`0b011`	`0b1101`	`0b1110`	`0b001`

MSR AMEVTYPER11_EL0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER11_EL0	`0b11`	`0b011`	`0b1101`	`0b1110`	`0b001`

B.8.3 AMEVTYPER12_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR12_EL0 counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

activity-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-111: AArch64_amevtyper12_el0 bit assignments

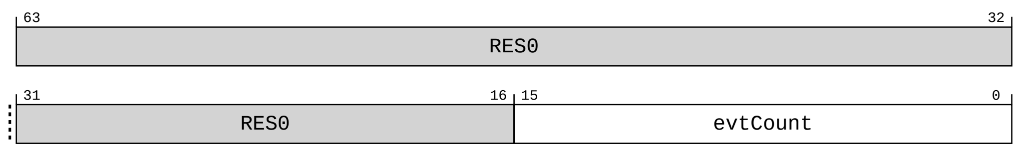


Table B-296: AMEVTYPER12_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	`0x0`
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AArch64-AMEVCNTR1<n>_EL0. It is **IMPLEMENTATION DEFINED** what values are supported by each counter. If software writes a value to this field which is not supported by the corresponding counter AArch64-AMEVCNTR1<n>_EL0, then: • It is UNPREDICTABLE which event will be counted. • The value read back is UNKNOWN. The event counted by AArch64-AMEVCNTR1<n>_EL0 might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED. If the corresponding counter AArch64-AMEVCNTR1<n>_EL0 is enabled, writes to this register have UNPREDICTABLE results.	

Access

MRS <Xt>, AMEVTYPER12_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER12_EL0	`0b11`	`0b011`	`0b1101`	`0b1110`	`0b010`

MSR AMEVTYPER12_EL0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER12_EL0	`0b11`	`0b011`	`0b1101`	`0b1110`	`0b010`

B.8.4 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

This register is available in all configurations.

Attributes

Width

64

Functional group

activity-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-112: AArch64_amcfgr_el0 bit assignments

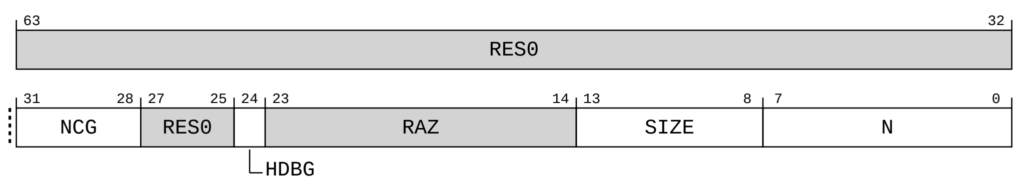


Table B-299: AMCFGR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	NCG	Defines the number of counter groups. The following value is specified for this product. **`0b0001`** Two counter groups are implemented	
[27:25]	RES0	Reserved	`0b000`
[24]	HDBG	Halt-on-debug supported. From Armv8, this feature must be supported, and so this bit is 1. **`0b1`** AArch64-AMCR_EL0.HDBG is read/write.	
[23:14]	RAZ	Reserved	

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 111111. **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.	
[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	

Access

MRS <Xt>, AMCFGR_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMCFGR_EL0	0b11	0b011	0b1101	0b0010	0b001

Accessibility

MRS <Xt>, AMCFGR_EL0

```
if PSTATE.EL == EL0 then
    if AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCFGR_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCFGR_EL0;
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCFGR_EL0;
elsif PSTATE.EL == EL3 then
    return AMCFGR_EL0;
```

B.8.5 AMCGCR_EL0, Activity Monitors Counter Group Configuration Register

Provides information on the number of activity monitor event counters implemented within each counter group.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

activity-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-113: AArch64_amcgcr_el0 bit assignments

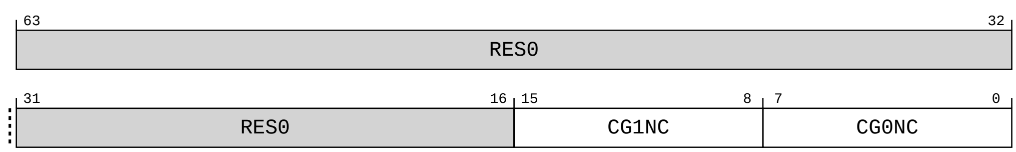


Table B-301: AMCGCR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	`0x0`
[15:8]	CG1NC	Counter Group 1 Number of Counters. The number of counters in the auxiliary counter group. In AMUv1, the permitted range of values is `0x0` to `0x10`. **`0b00000011`** Three counters in the auxiliary counter group	
[7:0]	CG0NC	Counter Group 0 Number of Counters. The number of counters in the architected counter group. In AMUv1, the value of this field is `0x4`. **`0b00000100`** Four Counters in the architected counter group	

Access

MRS <Xt>, AMCGCR_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMCGCR_EL0	0b11	0b011	0b1101	0b0010	0b010

Accessibility

MRS <Xt>, AMCGCR_EL0

```
if PSTATE.EL == EL0 then
    if AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCGCR_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCGCR_EL0;
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCGCR_EL0;
elsif PSTATE.EL == EL3 then
    return AMCGCR_EL0;
```

B.8.6 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

This register is available in all configurations.

Attributes

Width

64

Functional group

activity-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-114: AArch64_amevtyper00_el0 bit assignments

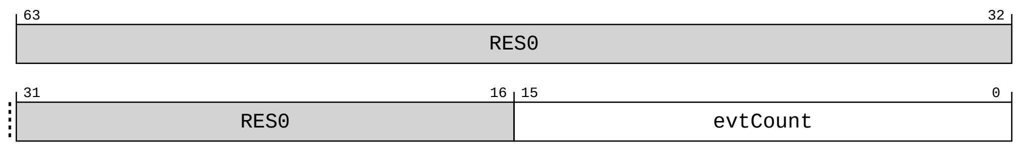


Table B-303: AMEVTYPER00_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR00_EL0. The value of this field is architecturally mandated for each architected counter.	

Access

MRS <Xt>, AMEVTYPER00_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER00_EL0	0b11	0b011	0b1101	0b0110	0b000

B.8.7 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

This register is available in all configurations.

Attributes

Width

64

Functional group

activity-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-115: AArch64_amevtyper01_el0 bit assignments

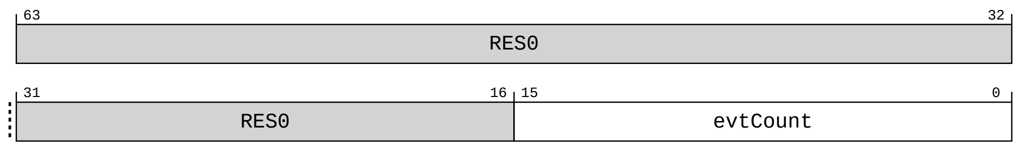


Table B-305: AMEVTYPER01_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR01_EL0. The value of this field is architecturally mandated for each architected counter.	

Access

MRS <Xt>, AMEVTYPER01_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER01_EL0	0b11	0b011	0b1101	0b0110	0b001

B.8.8 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

This register is available in all configurations.

Attributes

Width

64

Functional group

activity-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-116: AArch64_amevtyper02_el0 bit assignments

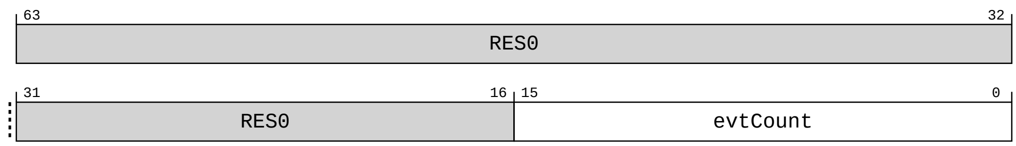


Table B-307: AMEVTYPER02_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR02_EL0. The value of this field is architecturally mandated for each architected counter.	

Access

MRS <Xt>, AMEVTYPER02_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER02_EL0	0b11	0b011	0b1101	0b0110	0b010

B.8.9 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

This register is available in all configurations.

Attributes

Width

64

Functional group

activity-monitors

Reset value

See individual bit resets.

Bit descriptions

Figure B-117: AArch64_amevtyper03_el0 bit assignments

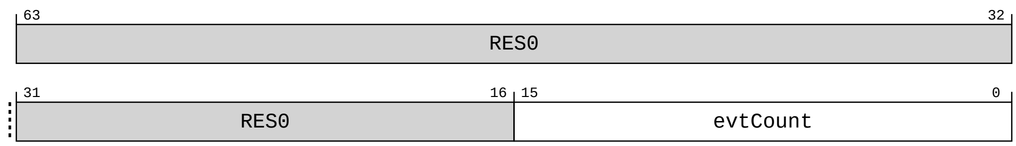


Table B-309: AMEVTYPER03_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	`0x0`
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR03_EL0. The value of this field is architecturally mandated for each architected counter.	

Access

MRS <Xt>, AMEVTYPER03_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER03_EL0	`0b11`	`0b011`	`0b1101`	`0b0110`	`0b011`

B.9 AArch64 trace registers

The summary table provides an overview of all implementation defined trace registers in the core. Individual register descriptions provide detailed information.

Table B-311: trace register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
TRCIDR8	2	C0	1	C0	6	See individual bit resets.	64-bit	ID Register 8
TRCIMSPEC0	2	C0	1	C0	7	See individual bit resets.	64-bit	IMP DEF Register 0
TRCIDR2	2	C0	1	C10	7	See individual bit resets.	64-bit	ID Register 2
TRCIDR3	2	C0	1	C11	7	See individual bit resets.	64-bit	ID Register 3
TRCIDR4	2	C0	1	C12	7	See individual bit resets.	64-bit	ID Register 4
TRCIDR5	2	C0	1	C13	7	See individual bit resets.	64-bit	ID Register 5
TRCIDR10	2	C0	1	C2	6	`0x0`	64-bit	ID Register 10
TRCIDR11	2	C0	1	C3	6	`0x0`	64-bit	ID Register 11
TRCIDR12	2	C0	1	C4	6	`0x0`	64-bit	ID Register 12
TRCIDR13	2	C0	1	C5	6	`0x0`	64-bit	ID Register 13
TRCIDR0	2	C0	1	C8	7	See individual bit resets.	64-bit	ID Register 0
TRCIDR1	2	C0	1	C9	7	See individual bit resets.	64-bit	ID Register 1
TRCCIDCVR0	2	C3	1	C0	0	See individual bit resets.	64-bit	Context Identifier Comparator Value Registers <n>

B.9.1 TRCIDR8, ID Register 8

Returns the maximum speculation depth of the instruction trace element stream.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-118: AArch64_trcidr8 bit assignments

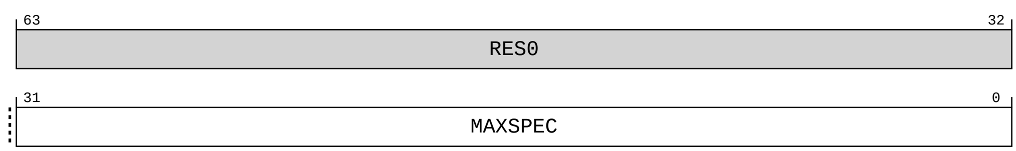


Table B-312: TRCIDR8 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:0]	MAXSPEC	Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of P0 elements in the trace element stream that can be speculative at any time. **0b00000000000000000000000000000000** No speculation in the trace element stream	

Access

MRS <Xt>, TRCIDR8

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR8	0b10	0b001	0b0000	0b0000	0b110

Accessibility

MRS <Xt>, TRCIDR8

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

```
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR8;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR8;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR8;
```

B.9.2 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

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-119: AArch64_trcimspec0 bit assignments

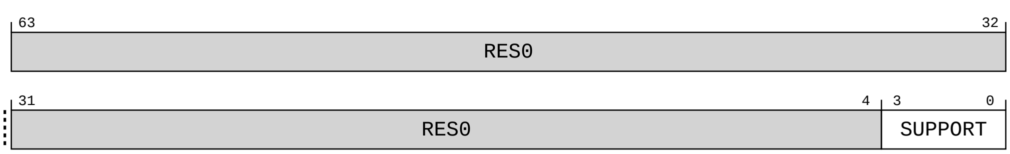


Table B-314: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[3:0]	SUPPORT	Indicates whether the implementation supports **IMPLEMENTATION DEFINED** features. **0b0000** No IMPLEMENTATION DEFINED features are supported.	

Access

MRS <Xt>, TRCIMSPEC0

<systemreg>	op0	op1	CRn	CRm	op2
TRCIMSPEC0	0b10	0b001	0b0000	0b0000	0b111

MSR TRCIMSPEC0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
TRCIMSPEC0	0b10	0b001	0b0000	0b0000	0b111

Accessibility

MRS <Xt>, TRCIMSPEC0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCIMSPECn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPEC0;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPEC0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPEC0;
```

MSR TRCIMSPEC0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRCIMSPECn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
```

```
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t];
```

B.9.3 TRCIDR2, ID Register 2

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-120: AArch64_trcidr2 bit assignments

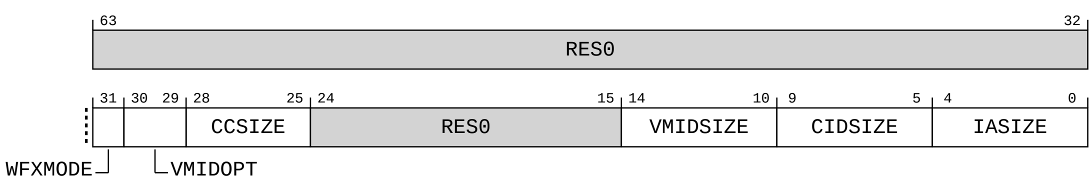


Table B-317: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31]	WFXMODE	Indicates whether WFI and WFE instructions are classified as P0 instructions: **`0b1`** WFI and WFE instructions are classified as P0 instructions.	

Bits	Name	Description	Reset
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. **0b10** Virtual context identifier selection not supported. AArch64-TRCCONFIGR.VMIDOPT is RES1.	
[28:25]	CCSIZE	Indicates the size of the cycle counter. **0b0000** The cycle counter is 12 bits in length.	
[24:15]	RES0	Reserved	0x0
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. **0b00100** 32-bit Virtual context identifier size.	
[9:5]	CIDSIZE	Indicates the Context identifier size. **0b00100** 32-bit Context identifier size.	
[4:0]	IASIZE	Virtual instruction address size. **0b01000** Maximum of 64-bit instruction address size.	

Access

MRS <Xt>, TRCIDR2

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR2	0b10	0b001	0b0000	0b1010	0b111

Accessibility

MRS <Xt>, TRCIDR2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR2;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR2;
```

B.9.4 TRCIDR3, ID Register 3

Returns the base architecture of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-121: AArch64_trcidr3 bit assignments

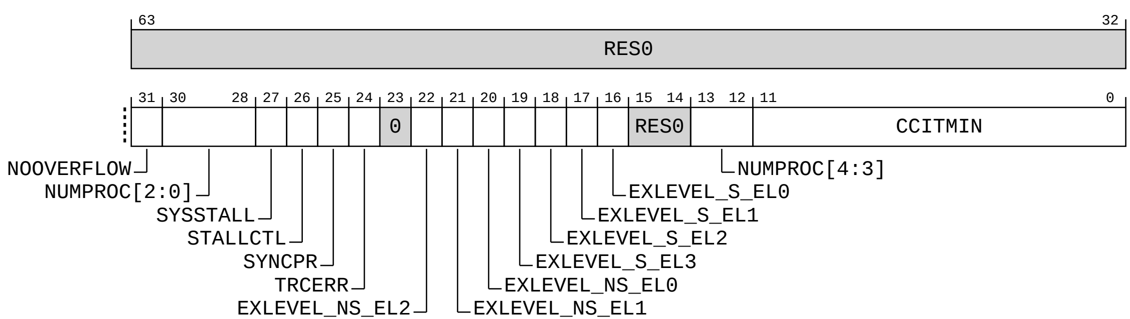


Table B-319: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. **0b00000** The trace unit can trace one PE.	
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **0b0** Stalling of the PE is not permitted.	

Bits	Name	Description	Reset
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **0b0** Stalling of the PE is not implemented.	
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0b0** AArch64-TRCSYNCPR is read-write so software can change the synchronization period.	
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **0b1** Forced tracing of System Error exceptions is implemented.	
[23]	RES0	Reserved	`0x0`
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 implemented. **0b1** Non-secure EL2 is implemented.	
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 implemented. **0b1** Non-secure EL1 is implemented.	
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 implemented. **0b1** Non-secure EL0 is implemented.	
[19]	EXLEVEL_S_EL3	Indicates if Secure EL3 implemented. **0b1** Secure EL3 is implemented.	
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 implemented. **0b1** Secure EL2 is implemented.	
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 implemented. **0b1** Secure EL1 is implemented.	
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 implemented. **0b1** Secure EL0 is implemented.	
[15:14]	RES0	Reserved	`0b00`
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in AArch64-TRCCCCTLR.THRESHOLD. If AArch64-TRCIDR0.TRCCCI == 1 then the minimum value of this field is `0x001`. If AArch64-TRCIDR0.TRCCCI == 0 then this field is zero. **0b000000000100**	

Access

MRS <Xt>, TRCIDR3

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR3	0b10	0b001	0b0000	0b1011	0b111

Accessibility

MRS <Xt>, TRCIDR3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR3;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR3;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR3;
```

B.9.5 TRCIDR4, ID Register 4

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-122: AArch64_trcidr4 bit assignments

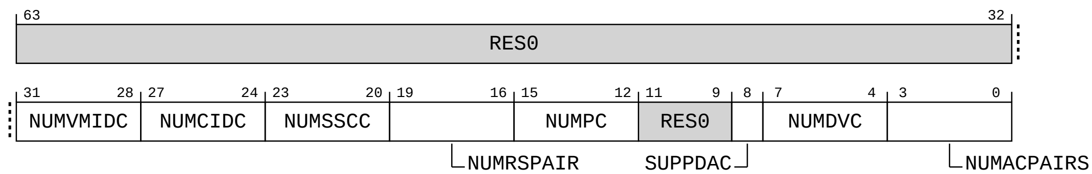


Table B-321: TRCIDR4 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. **`0b0001`** The implementation has one Virtual Context Identifier Comparator.	
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **`0b0001`** The implementation has one Context Identifier Comparator.	
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. **`0b0001`** The implementation has one Single-shot Comparator Control.	
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **`0b0111`** The implementation has eight resource selector pairs.	
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **`0b0000`** No PE Comparator Inputs are available.	
[11:9]	RES0	Reserved	`0b000`
[8]	SUPPDAC	Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **`0b0`** Data address comparisons not implemented.	
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **`0b0000`** No data value comparators implemented.	
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **`0b0100`** The implementation has four Address Comparator pairs.	

Access

MRS <Xt>, TRCIDR4

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR4	0b10	0b001	0b0000	0b1100	0b111

Accessibility

MRS <Xt>, TRCIDR4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;
```

B.9.6 TRCIDR5, ID Register 5

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-123: AArch64_trcidr5 bit assignments

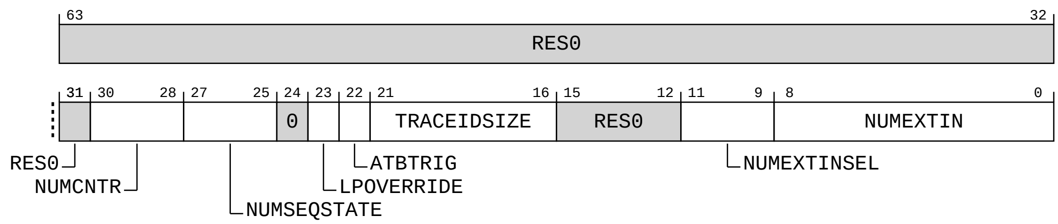


Table B-323: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	`0x0`
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. **`0b010`** Two Counters implemented.	
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **`0b100`** Four Sequencer states are implemented.	
[24]	RES0	Reserved	`0b0`
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **`0b0`** The trace unit does not support Low-power Override Mode.	
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **`0b1`** The implementation supports ATB triggers.	
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **`0b000111`** The implementation supports a 7-bit trace ID.	
[15:12]	RES0	Reserved	`0b0000`
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. **`0b100`** 4 External Input Selector resources are available.	
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **`0b111111111`** Unified PMU event selection.	

Access

MRS <Xt>, TRCIDR5

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR5	`0b10`	`0b001`	`0b0000`	`0b1101`	`0b111`

Accessibility

MRS <Xt>, TRCIDR5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;
```

B.9.7 TRCIDR10, ID Register 10

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

0x0

Bit descriptions

Figure B-124: AArch64_trcidr10 bit assignments

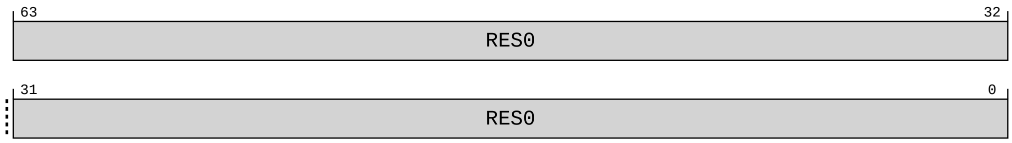


Table B-325: TRCIDR10 bit descriptions

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

Access

MRS <Xt>, TRCIDR10

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR10	0b10	0b001	0b0000	0b0010	0b110

Accessibility

MRS <Xt>, TRCIDR10

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
```

B.9.8 TRCIDR11, ID Register 11

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

`0x0`

Bit descriptions

Figure B-125: AArch64_trcidr11 bit assignments

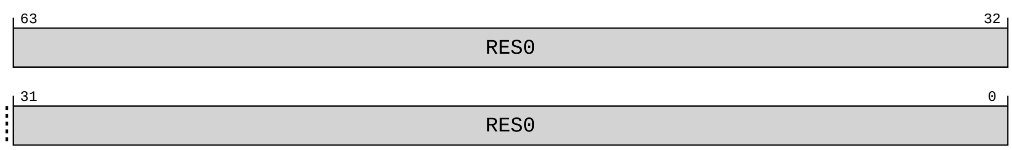


Table B-327: TRCIDR11 bit descriptions

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

Access

MRS <Xt>, TRCIDR11

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR11	`0b10`	`0b001`	`0b0000`	`0b0011`	`0b110`

Accessibility

MRS <Xt>, TRCIDR11

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    else
        return TRCIDR11;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR11;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR11;
```

B.9.9 TRCIDR12, ID Register 12

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

0x0

Bit descriptions

Figure B-126: AArch64_trcidr12 bit assignments

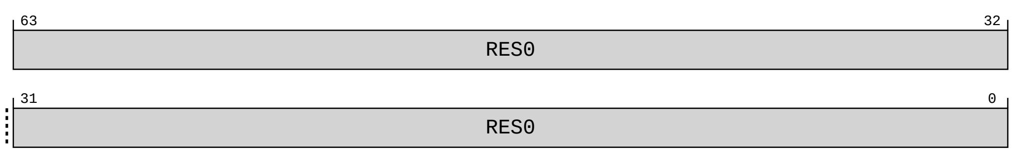


Table B-329: TRCIDR12 bit descriptions

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

Access

MRS <Xt>, TRCIDR12

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR12	0b10	0b001	0b0000	0b0100	0b110

Accessibility

MRS <Xt>, TRCIDR12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR12;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR12;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR12;
```

B.9.10 TRCIDR13, ID Register 13

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

`0x0`

Bit descriptions

Figure B-127: AArch64_trcidr13 bit assignments

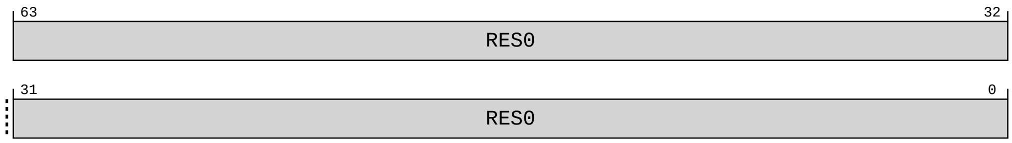


Table B-331: TRCIDR13 bit descriptions

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

Access

MRS <Xt>, TRCIDR13

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR13	0b10	0b001	0b0000	0b0101	0b110

Accessibility

MRS <Xt>, TRCIDR13

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
```

B.9.11 TRCIDR0, ID Register 0

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-128: AArch64_trcidr0 bit assignments

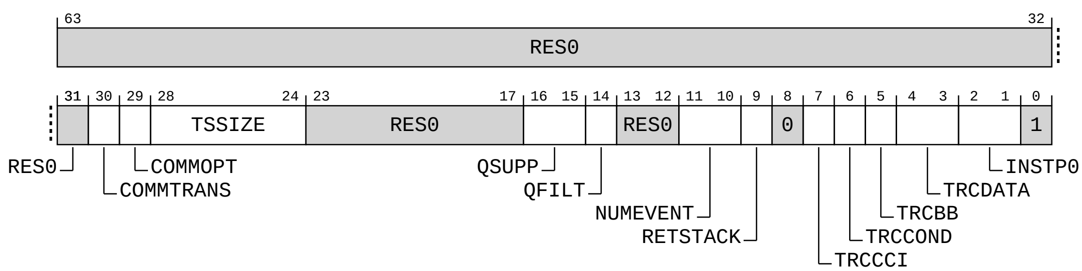


Table B-333: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	`0x0`
[30]	COMMTRANS	Transaction Start element behavior. **`0b0`** Transaction Start elements are P0 elements.	
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **`0b1`** Commit mode 1.	
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. **`0b01000`** Global timestamping implemented with a 64-bit timestamp value.	
[23:17]	RES0	Reserved	`0b0000000`
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **`0b00`** Q element support is not implemented.	

Bits	Name	Description	Reset
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	
[13:12]	RES0	Reserved	0b00
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **0b11** The trace unit supports 4 ETEEvents.	
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **0b1** Return stack implemented.	
[8]	RES0	Reserved	0b0
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **0b1** Cycle counting implemented.	
[6]	TRCCOND	Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b0** Conditional instruction tracing not implemented.	
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. **0b1** Branch broadcasting implemented.	
[4:3]	TRCDATA	Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Tracing of data addresses and data values is not implemented.	
[2:1]	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.	
[0]	RES1	Reserved	0b1

Access

MRS <Xt>, TRCIDR0

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR0	0b10	0b001	0b0000	0b1000	0b111

Accessibility

MRS <Xt>, TRCIDR0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
```

```
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR0;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR0;
```

B.9.12 TRCIDR1, ID Register 1

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-129: AArch64_trcidr1 bit assignments

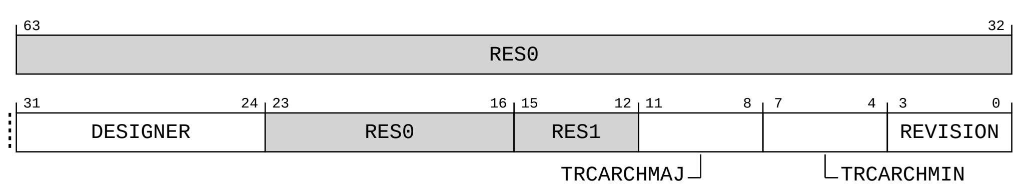


Table B-335: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[31:24]	DESIGNER	Indicates which company designed the trace unit. The permitted values of this field are the same as AArch64-MIDR_EL1.Implementer. **0b01000001** Arm Limited	
[23:16]	RES0	Reserved	0b00000000
[15:12]	RES1	Reserved	0b1111
[11:8]	TRCARCHMAJ	Major architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH.	
[7:4]	TRCARCHMIN	Minor architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH.	
[3:0]	REVISION	Implementation revision that identifies the revision of the trace and OS Lock registers. **0b0000** Revision 0	

Access

MRS <Xt>, TRCIDR1

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR1	0b10	0b001	0b0000	0b1001	0b111

Accessibility

MRS <Xt>, TRCIDR1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
```

B.9.13 TRCCIDCVR0, Context Identifier Comparator Value Registers <n>

Contains a Context identifier value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-130: AArch64_trccidcvr0 bit assignments

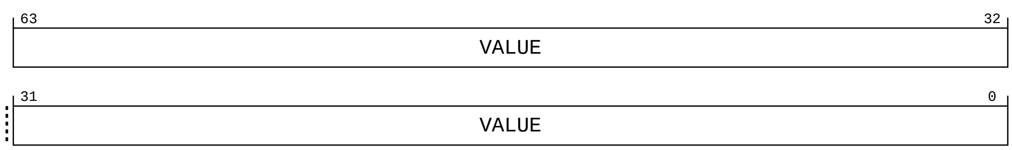


Table B-337: TRCCIDCVR0 bit descriptions

Bits	Name	Description	Reset
[63:0]	VALUE	Context identifier value. The width of this field is indicated by AArch64-TRCIDR2.CIDSIZE. Unimplemented bits are RES0. After a PE Reset, the trace unit assumes that the Context identifier is zero until the PE updates the Context identifier.	

Access

MRS <Xt>, TRCCIDCVR0

<systemreg>	op0	op1	CRn	CRm	op2
TRCCIDCVR0	0b10	0b001	0b0011	0b0000	0b000

MSR TRCCIDCVR0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
TRCCIDCVR0	0b10	0b001	0b0011	0b0000	0b000

B.10 AArch64 MPAM registers

The summary table provides an overview of all implementation defined MPAM registers in the core. Individual register descriptions provide detailed information.

Table B-340: MPAM register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
MPAMVPMV_EL2	3	C10	4	C4	1	See individual bit resets.	64-bit	MPAM Virtual Partition Mapping Valid Register
MPAMVPM0_EL2	3	C10	4	C6	0	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 0
MPAMVPM1_EL2	3	C10	4	C6	1	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 1
MPAMVPM2_EL2	3	C10	4	C6	2	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 2
MPAMVPM3_EL2	3	C10	4	C6	3	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 3
MPAMVPM4_EL2	3	C10	4	C6	4	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 4
MPAMVPM5_EL2	3	C10	4	C6	5	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 5
MPAMVPM6_EL2	3	C10	4	C6	6	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 6
MPAMVPM7_EL2	3	C10	4	C6	7	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 7

B.10.1 MPAMVPMV_EL2, MPAM Virtual Partition Mapping Valid Register

Valid bits for virtual PARTID mapping entries. Each bit m corresponds to virtual PARTID mapping entry m in the MPAMVPM<n>_EL2 registers where n = m >> 2.

Configurations

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

Attributes

Width

64

Functional group

mpam

Reset value

See individual bit resets.

Bit descriptions

Figure B-131: AArch64_mpamvpmv_el2 bit assignments

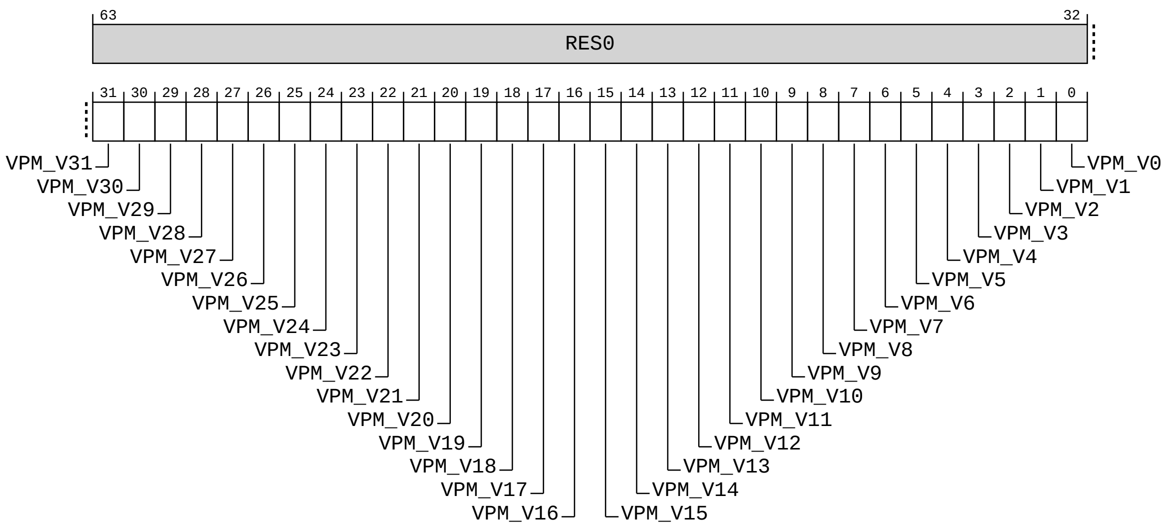


Table B-341: MPAMVPMV_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31]	VPM_V31	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[30]	VPM_V30	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[29]	VPM_V29	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[28]	VPM_V28	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[27]	VPM_V27	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[26]	VPM_V26	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[25]	VPM_V25	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[24]	VPM_V24	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[23]	VPM_V23	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[22]	VPM_V22	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[21]	VPM_V21	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[20]	VPM_V20	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[19]	VPM_V19	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[18]	VPM_V18	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[17]	VPM_V17	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[16]	VPM_V16	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[15]	VPM_V15	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[14]	VPM_V14	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[13]	VPM_V13	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[12]	VPM_V12	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[11]	VPM_V11	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[10]	VPM_V10	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	

Bits	Name	Description	Reset
[9]	VPM_V9	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[8]	VPM_V8	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[7]	VPM_V7	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[6]	VPM_V6	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[5]	VPM_V5	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[4]	VPM_V4	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[3]	VPM_V3	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[2]	VPM_V2	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[1]	VPM_V1	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[0]	VPM_V0	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	

Access

MRS <Xt>, MPAMVPMV_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPMV_EL2	0b11	0b100	0b1010	0b0100	0b001

MSR MPAMVPMV_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPMV_EL2	0b11	0b100	0b1010	0b0100	0b001

Accessibility

MRS <Xt>, MPAMVPMV_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x938];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMVPMV_EL2;
elsif PSTATE.EL == EL3 then
    return MPAMVPMV_EL2;
```

MSR MPAMVPMV_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
```

```
        NVMem[0x938] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPMV_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPMV_EL2 = X[t];
```

B.10.2 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

mpam

Reset value

See individual bit resets.

Bit descriptions

Figure B-132: AArch64_mpamvpm0_el2 bit assignments

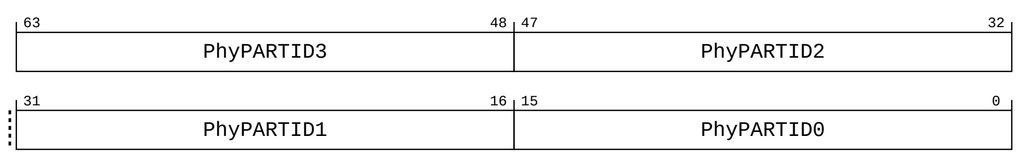


Table B-344: MPAMVPM0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID3	Virtual PARTID Mapping Entry for virtual PARTID 3. PhyPARTID3 gives the mapping of virtual PARTID 3 to a physical PARTID.	
[47:32]	PhyPARTID2	Virtual PARTID Mapping Entry for virtual PARTID 2. PhyPARTID2 gives the mapping of virtual PARTID 2 to a physical PARTID.	
[31:16]	PhyPARTID1	Virtual PARTID Mapping Entry for virtual PARTID 1. PhyPARTID1 gives the mapping of virtual PARTID 1 to a physical PARTID.	
[15:0]	PhyPARTID0	Virtual PARTID Mapping Entry for virtual PARTID 0. PhyPARTID0 gives the mapping of virtual PARTID 0 to a physical PARTID.	

Access

MRS <Xt>, MPAMVPM0_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM0_EL2	0b11	0b100	0b1010	0b0110	0b000

MSR MPAMVPM0_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM0_EL2	0b11	0b100	0b1010	0b0110	0b000

Accessibility

MRS <Xt>, MPAMVPM0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x940];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return 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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x940] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM0_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPM0_EL2 = X[t];
```

B.10.3 MPAMVPM1_EL2, MPAM Virtual PARTID Mapping Register 1

MPAMVPM1_EL2 provides mappings from virtual PARTIDs 4 - 7 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented AArch64-MPAMVPM0_EL2 to AArch64-MPAMVPM7_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

mpam

Reset value

See individual bit resets.

Bit descriptions

Figure B-133: AArch64_mpamvpm1_el2 bit assignments

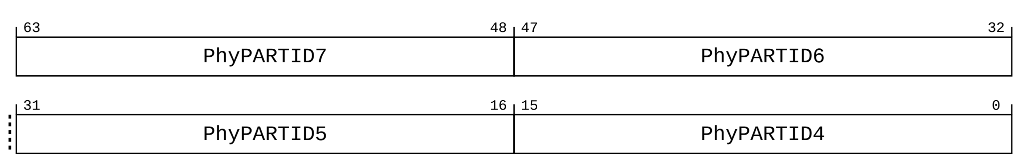


Table B-347: MPAMVPM1_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID7	Virtual PARTID Mapping Entry for virtual PARTID 7. PhyPARTID7 gives the mapping of virtual PARTID 7 to a physical PARTID.	
[47:32]	PhyPARTID6	Virtual PARTID Mapping Entry for virtual PARTID 6. PhyPARTID6 gives the mapping of virtual PARTID 6 to a physical PARTID.	
[31:16]	PhyPARTID5	Virtual PARTID Mapping Entry for virtual PARTID 5. PhyPARTID5 gives the mapping of virtual PARTID 5 to a physical PARTID.	
[15:0]	PhyPARTID4	Virtual PARTID Mapping Entry for virtual PARTID 4. PhyPARTID4 gives the mapping of virtual PARTID 4 to a physical PARTID.	

Access

MRS <Xt>, MPAMVPM1_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM1_EL2	0b11	0b100	0b1010	0b0110	0b001

MSR MPAMVPM1_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM1_EL2	0b11	0b100	0b1010	0b0110	0b001

Accessibility

MRS <Xt>, MPAMVPM1_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x948];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return 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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x948] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM1_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPM1_EL2 = X[t];
```

B.10.4 MPAMVPM2_EL2, MPAM Virtual PARTID Mapping Register 2

MPAMVPM2_EL2 provides mappings from virtual PARTIDs 8 - 11 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented AArch64-MPAMVPM0_EL2 to AArch64-MPAMVPM7_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

mpam

Reset value

See individual bit resets.

Bit descriptions

Figure B-134: AArch64_mpamvpm2_el2 bit assignments

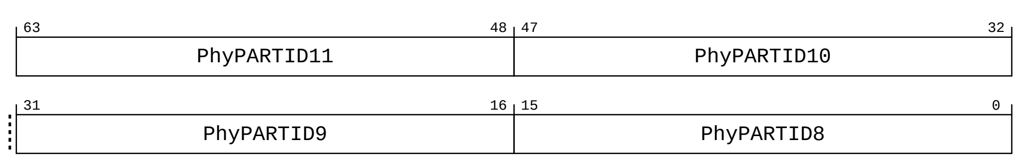


Table B-350: MPAMVPM2_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID11	Virtual PARTID Mapping Entry for virtual PARTID 11. PhyPARTID11 gives the mapping of virtual PARTID 11 to a physical PARTID.	
[47:32]	PhyPARTID10	Virtual PARTID Mapping Entry for virtual PARTID 10. PhyPARTID10 gives the mapping of virtual PARTID 10 to a physical PARTID.	
[31:16]	PhyPARTID9	Virtual PARTID Mapping Entry for virtual PARTID 9. PhyPARTID9 gives the mapping of virtual PARTID 9 to a physical PARTID.	
[15:0]	PhyPARTID8	Virtual PARTID Mapping Entry for virtual PARTID 8. PhyPARTID8 gives the mapping of virtual PARTID 8 to a physical PARTID.	

Access

MRS <Xt>, MPAMVPM2_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM2_EL2	0b11	0b100	0b1010	0b0110	0b010

MSR MPAMVPM2_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM2_EL2	0b11	0b100	0b1010	0b0110	0b010

Accessibility

MRS <Xt>, MPAMVPM2_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x950];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMVPM2_EL2;
elsif PSTATE.EL == EL3 then
    return MPAMVPM2_EL2;
```

MSR MPAMVPM2_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x950] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM2_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPM2_EL2 = X[t];
```

B.10.5 MPAMVPM3_EL2, MPAM Virtual PARTID Mapping Register 3

MPAMVPM3_EL2 provides mappings from virtual PARTIDs 12 - 15 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

mpam

Reset value

See individual bit resets.

Bit descriptions

Figure B-135: AArch64_mpamvpm3_el2 bit assignments

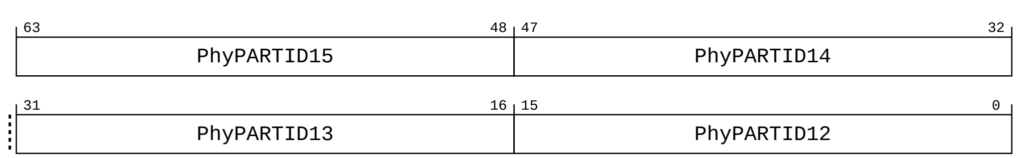


Table B-353: MPAMVPM3_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID15	Virtual PARTID Mapping Entry for virtual PARTID 15. PhyPARTID15 gives the mapping of virtual PARTID 15 to a physical PARTID.	
[47:32]	PhyPARTID14	Virtual PARTID Mapping Entry for virtual PARTID 14. PhyPARTID14 gives the mapping of virtual PARTID 14 to a physical PARTID.	
[31:16]	PhyPARTID13	Virtual PARTID Mapping Entry for virtual PARTID 13. PhyPARTID13 gives the mapping of virtual PARTID 13 to a physical PARTID.	
[15:0]	PhyPARTID12	Virtual PARTID Mapping Entry for virtual PARTID 12. PhyPARTID12 gives the mapping of virtual PARTID 12 to a physical PARTID.	

Access

MRS <Xt>, MPAMVPM3_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM3_EL2	0b11	0b100	0b1010	0b0110	0b011

MSR MPAMVPM3_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM3_EL2	0b11	0b100	0b1010	0b0110	0b011

Accessibility

MRS <Xt>, MPAMVPM3_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x958];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMVPM3_EL2;
elsif PSTATE.EL == EL3 then
    return MPAMVPM3_EL2;
```

MSR MPAMVPM3_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x958] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM3_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPM3_EL2 = X[t];
```

B.10.6 MPAMVPM4_EL2, MPAM Virtual PARTID Mapping Register 4

MPAMVPM4_EL2 provides mappings from virtual PARTIDs 16 - 19 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

mpam

Reset value

See individual bit resets.

Bit descriptions

Figure B-136: AArch64_mpamvpm4_el2 bit assignments

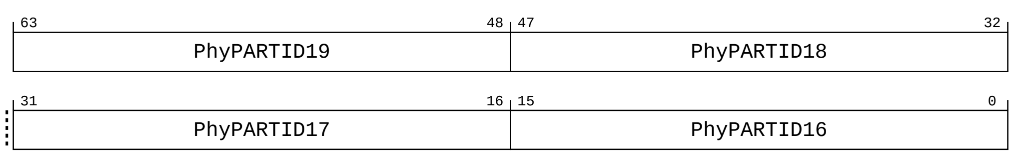


Table B-356: MPAMVPM4_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID19	Virtual PARTID Mapping Entry for virtual PARTID 19. PhyPARTID19 gives the mapping of virtual PARTID 19 to a physical PARTID.	
[47:32]	PhyPARTID18	Virtual PARTID Mapping Entry for virtual PARTID 18. PhyPARTID18 gives the mapping of virtual PARTID 18 to a physical PARTID.	
[31:16]	PhyPARTID17	Virtual PARTID Mapping Entry for virtual PARTID 17. PhyPARTID17 gives the mapping of virtual PARTID 17 to a physical PARTID.	
[15:0]	PhyPARTID16	Virtual PARTID Mapping Entry for virtual PARTID 16. PhyPARTID16 gives the mapping of virtual PARTID 16 to a physical PARTID.	

Access

MRS <Xt>, MPAMVPM4_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM4_EL2	0b11	0b100	0b1010	0b0110	0b100

MSR MPAMVPM4_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM4_EL2	0b11	0b100	0b1010	0b0110	0b100

Accessibility

MRS <Xt>, MPAMVPM4_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x960];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMVPM4_EL2;
elsif PSTATE.EL == EL3 then
    return MPAMVPM4_EL2;
```

MSR MPAMVPM4_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x960] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM4_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPM4_EL2 = X[t];
```

B.10.7 MPAMVPM5_EL2, MPAM Virtual PARTID Mapping Register 5

MPAMVPM5_EL2 provides mappings from virtual PARTIDs 20 - 23 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

mpam

Reset value

See individual bit resets.

Bit descriptions

Figure B-137: AArch64_mpamvpm5_el2 bit assignments

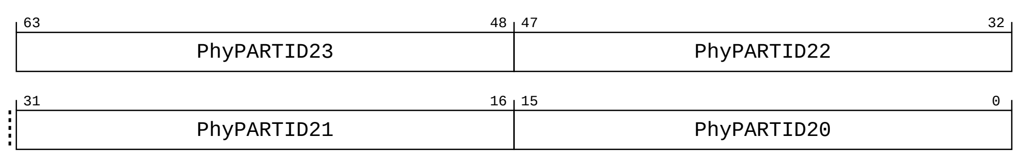


Table B-359: MPAMVPM5_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID23	Virtual PARTID Mapping Entry for virtual PARTID 23. PhyPARTID23 gives the mapping of virtual PARTID 23 to a physical PARTID.	
[47:32]	PhyPARTID22	Virtual PARTID Mapping Entry for virtual PARTID 22. PhyPARTID22 gives the mapping of virtual PARTID 22 to a physical PARTID.	
[31:16]	PhyPARTID21	Virtual PARTID Mapping Entry for virtual PARTID 21. PhyPARTID21 gives the mapping of virtual PARTID 21 to a physical PARTID.	
[15:0]	PhyPARTID20	Virtual PARTID Mapping Entry for virtual PARTID 20. PhyPARTID20 gives the mapping of virtual PARTID 20 to a physical PARTID.	

Access

MRS <Xt>, MPAMVPM5_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM5_EL2	0b11	0b100	0b1010	0b0110	0b101

MSR MPAMVPM5_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM5_EL2	0b11	0b100	0b1010	0b0110	0b101

Accessibility

MRS <Xt>, MPAMVPM5_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x968];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMVPM5_EL2;
elsif PSTATE.EL == EL3 then
    return MPAMVPM5_EL2;
```

MSR MPAMVPM5_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x968] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM5_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPM5_EL2 = X[t];
```

B.10.8 MPAMVPM6_EL2, MPAM Virtual PARTID Mapping Register 6

MPAMVPM6_EL2 provides mappings from virtual PARTIDs 24 - 27 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

mpam

Reset value

See individual bit resets.

Bit descriptions

Figure B-138: AArch64_mpamvpm6_el2 bit assignments

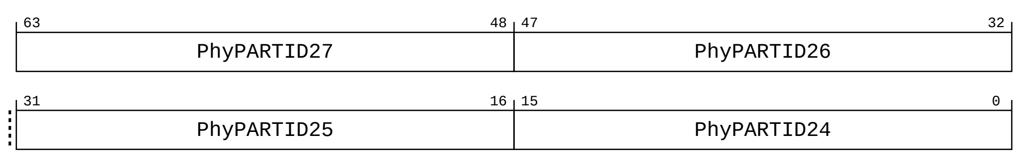


Table B-362: MPAMVPM6_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID27	Virtual PARTID Mapping Entry for virtual PARTID 27. PhyPARTID27 gives the mapping of virtual PARTID 27 to a physical PARTID.	
[47:32]	PhyPARTID26	Virtual PARTID Mapping Entry for virtual PARTID 26. PhyPARTID26 gives the mapping of virtual PARTID 26 to a physical PARTID.	
[31:16]	PhyPARTID25	Virtual PARTID Mapping Entry for virtual PARTID 25. PhyPARTID25 gives the mapping of virtual PARTID 25 to a physical PARTID.	
[15:0]	PhyPARTID24	Virtual PARTID Mapping Entry for virtual PARTID 24. PhyPARTID24 gives the mapping of virtual PARTID 24 to a physical PARTID.	

Access

MRS <Xt>, MPAMVPM6_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM6_EL2	0b11	0b100	0b1010	0b0110	0b110

MSR MPAMVPM6_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM6_EL2	0b11	0b100	0b1010	0b0110	0b110

Accessibility

MRS <Xt>, MPAMVPM6_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x970];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMVPM6_EL2;
elsif PSTATE.EL == EL3 then
    return MPAMVPM6_EL2;
```

MSR MPAMVPM6_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x970] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM6_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPM6_EL2 = X[t];
```

B.10.9 MPAMVPM7_EL2, MPAM Virtual PARTID Mapping Register 7

MPAMVPM7_EL2 provides mappings from virtual PARTIDs 28 - 31 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for AArch64-MPAM0_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

mpam

Reset value

See individual bit resets.

Bit descriptions

Figure B-139: AArch64_mpamvpm7_el2 bit assignments

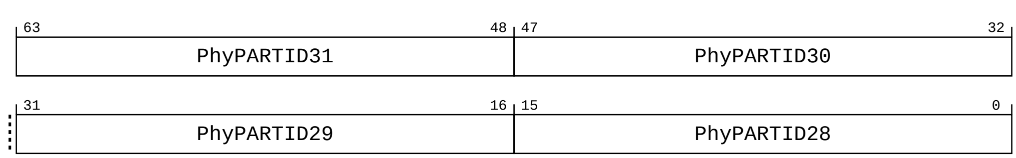


Table B-365: MPAMVPM7_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID31	Virtual PARTID Mapping Entry for virtual PARTID 31. PhyPARTID31 gives the mapping of virtual PARTID 31 to a physical PARTID.	
[47:32]	PhyPARTID30	Virtual PARTID Mapping Entry for virtual PARTID 30. PhyPARTID30 gives the mapping of virtual PARTID 30 to a physical PARTID.	
[31:16]	PhyPARTID29	Virtual PARTID Mapping Entry for virtual PARTID 29. PhyPARTID29 gives the mapping of virtual PARTID 29 to a physical PARTID.	
[15:0]	PhyPARTID28	Virtual PARTID Mapping Entry for virtual PARTID 28. PhyPARTID28 gives the mapping of virtual PARTID 28 to a physical PARTID.	

Access

MRS <Xt>, MPAMVPM7_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM7_EL2	0b11	0b100	0b1010	0b0110	0b111

MSR MPAMVPM7_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM7_EL2	0b11	0b100	0b1010	0b0110	0b111

Accessibility

MRS <Xt>, MPAMVPM7_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x978];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMVPM7_EL2;
elsif PSTATE.EL == EL3 then
    return MPAMVPM7_EL2;
```

MSR MPAMVPM7_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x978] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM7_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    MPAMVPM7_EL2 = X[t];
```

B.11 AArch64 RAS registers

The summary table provides an overview of all implementation defined ras registers in the core. Individual register descriptions provide detailed information.

Table B-368: ras register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
ERRIDR_EL1	3	C5	0	C3	0	See individual bit resets.	64-bit	Error Record ID Register
ERRSELR_EL1	3	C5	0	C3	1	See individual bit resets.	64-bit	Error Record Select Register
ERXFR_EL1	3	C5	0	C4	0	See individual bit resets.	64-bit	Selected Error Record Feature Register
ERXCTLR_EL1	3	C5	0	C4	1	`0x0`	64-bit	Selected Error Record Control Register
ERXSTATUS_EL1	3	C5	0	C4	2	`0x0`	64-bit	Selected Error Record Primary Status Register
ERXADDR_EL1	3	C5	0	C4	3	See individual bit resets.	64-bit	Selected Error Record Address Register
ERXPFGF_EL1	3	C5	0	C4	4	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Feature register
ERXPFGCTL_EL1	3	C5	0	C4	5	`0x0`	64-bit	Selected Pseudo-fault Generation Control register
ERXPFGCDN_EL1	3	C5	0	C4	6	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Countdown register
ERXMISC0_EL1	3	C5	0	C5	0	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 0
ERXMISC1_EL1	3	C5	0	C5	1	`0x0`	64-bit	Selected Error Record Miscellaneous Register 1
ERXMISC2_EL1	3	C5	0	C5	2	`0x0`	64-bit	Selected Error Record Miscellaneous Register 2
ERXMISC3_EL1	3	C5	0	C5	3	`0x0`	64-bit	Selected Error Record Miscellaneous Register 3

B.11.1 ERRIDR_EL1, Error Record ID Register

Defines the highest numbered index of the error records that can be accessed through the Error Record System registers.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure B-140: AArch64_erridr_el1 bit assignments

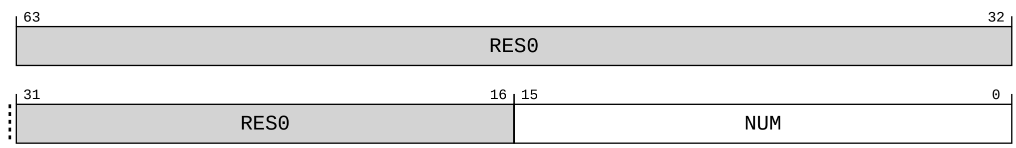


Table B-369: ERRIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	`0x0`
[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.	

Access

MRS <Xt>, ERRIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERRIDR_EL1	`0b11`	`0b000`	`0b0101`	`0b0011`	`0b000`

Accessibility

MRS <Xt>, ERRIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERRIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRIDR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRIDR_EL1;
elsif PSTATE.EL == EL3 then
    return ERRIDR_EL1;
```

B.11.2 ERRSELR_EL1, Error Record Select Register

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

Configurations

If AArch64-ERRIDR_EL1 indicates that zero error records are implemented, then it is IMPLEMENTATION DEFINED whether ERRSELR_EL1 is UNDEFINED or RES0.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure B-141: AArch64_errselr_el1 bit assignments

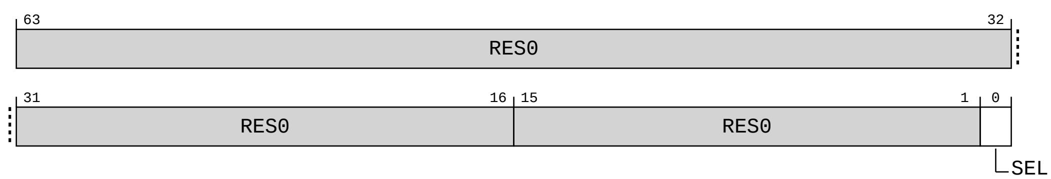


Table B-371: ERRSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	0x0
[0]	SEL	**0b0** Selects record 0, containing errors from DSU RAMs **0b1** Selects record 1, containing errors from Core RAMs	

Access

MRS <Xt>, ERRSELR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERRSELR_EL1	0b11	0b000	0b0101	0b0011	0b001

MSR ERRSELR_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERRSELR_EL1	0b11	0b000	0b0101	0b0011	0b001

Accessibility

MRS <Xt>, ERRSELR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERRSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRSELR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRSELR_EL1;
elsif PSTATE.EL == EL3 then
    return ERRSELR_EL1;
```

MSR ERRSELR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERRSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERRSELR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERRSELR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERRSELR_EL1 = X[t];
```

B.11.3 ERXFR_EL1, Selected Error Record Feature Register

Accesses ext-ERR<n>FR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure B-142: AArch64_erxfr_el1 bit assignments

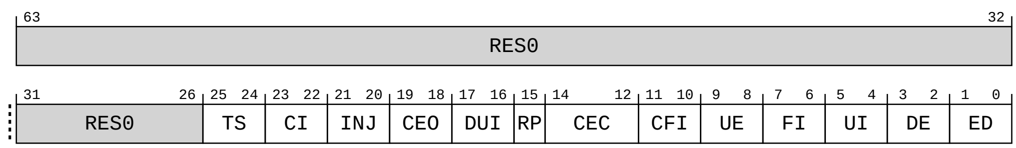


Table B-374: ERXFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:26]	RES0	Reserved	0b000000
[25:24]	TS	Timestamp Extension. Indicates whether, for each error record <m> owned by this node, rERXMISC3_EL1 is used as the timestamp register, and, if it is, the timebase used by the timestamp. **0b00** The node does not support a timestamp register.	
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented. **0b00** Does not support the critical error interrupt. ERXCTLR_EL1.CI is RES0.	
[21:20]	INJ	Fault Injection Extension. Indicates whether the RAS Common Fault Injection Model Extension is implemented. **0b01** The node implements the RAS Common Fault Injection Model Extension. See ERXPFGF_EL1 for more information.	
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior when a second Corrected error is detected after a first Corrected error has been recorded by an error record <m> owned by the node. **0b00** Counts Corrected errors if a counter is implemented. Keeps the previous error syndrome. If the counter overflows, or no counter is implemented, then ERXSTATUS_EL1.OF is set to 0b1.	
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the control for enabling error recovery interrupts on deferred errors are implemented. **0b00** Does not support the control for enabling error recovery interrupts on deferred errors. ERXCTLR_EL1.DUI is RES0.	
[15]	RP	Repeat counter. Indicates whether the node implements the repeat Corrected error counter in ERXMISC0_EL1 for each error record <m> owned by the node that implements the standard Corrected error counter. **0b1** A first (repeat) counter and a second (other) counter are implemented. The repeat counter is the same size as the primary error counter.	

Bits	Name	Description	Reset
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter (CE counter) mechanisms in ERXMISC0_EL1 for each error record <m> owned by the node that can record countable errors. **0b010** Implements an 8-bit Corrected error counter in ERXMISC0_EL1[39:32].	
[11:10]	CFI	Fault handling interrupt for corrected errors. Indicates whether the control for enabling fault handling interrupts on corrected errors are implemented. **0b10** Control for enabling fault handling interrupts on corrected errors is supported and controllable using ERXCTLR_EL1.CFI.	
[9:8]	UE	In-band uncorrected error reporting. Indicates whether the in-band uncorrected error reporting (External Aborts) and associated controls are implemented. **0b01** In-band uncorrected error reporting (External Aborts) is supported and always enabled. ERXCTLR_EL1.UE is RES0.	
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented. **0b10** Fault handling interrupt is supported and controllable using ERXCTLR_EL1.FI.	
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented. **0b10** Error handling interrupt is supported and controllable using ERXCTLR_EL1.UI.	
[3:2]	DE	**0b00**	
[1:0]	ED	Error reporting and logging. Indicates whether error record <n> is the first record owned the node, and, if so, whether it implements the controls for enabling and disabling error reporting and logging. **0b10** Error reporting and logging is controllable using ERXCTLR_EL1.ED.	

Access

MRS <Xt>, ERXFR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXFR_EL1	0b11	0b000	0b0101	0b0100	0b000

Accessibility

MRS <Xt>, ERXFR_EL1

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

```
        return ERXFR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXFR_EL1;
elsif PSTATE.EL == EL3 then
    return ERXFR_EL1;
```

B.11.4 ERXCTLR_EL1, Selected Error Record Control Register

Accesses ext-ERR<n>CTLR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

`0x0`

Bit descriptions

Figure B-143: AArch64_erxctlr_el1 bit assignments

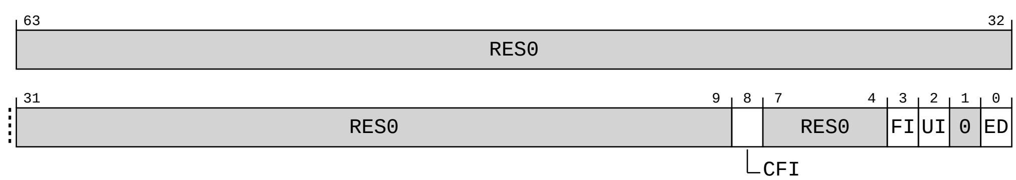


Table B-376: ERXCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:9]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[8]	CFI	Fault handling interrupt for Corrected errors enable. This control applies to errors arising from both reads and writes. The fault handling interrupt is generated when one of the standard CE counters on ERXMISC0_EL1 overflows and the overflow bit is set. The possible values are: **0b0** Fault handling interrupt not generated for Corrected errors. **0b1** Fault handling interrupt generated for Corrected errors. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error. Cold reset only. Unaffected by Warm reset	`0x0`
[7:4]	RES0	Reserved	`0b0000`
[3]	FI	Fault handling interrupt enable. This control applies to errors arising from both reads and writes. The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. The possible values are: **0b0** Fault handling interrupt disabled. **0b1** Fault handling interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error. Cold reset only. Unaffected by Warm reset	`0b0`
[2]	UI	Uncorrected error recovery interrupt enable. This control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred. **0b0** Error recovery interrupt disabled. **0b1** Error recovery interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error. Cold reset only. Unaffected by Warm reset	`0b0`
[1]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[0]	ED	Error Detection and correction enable. The possible values are: **0b0** Error detection and correction disabled. **0b1** Error detection and correction enabled. Cold reset only. Unaffected by Warm reset	0b0

Access

MRS <Xt>, ERXCTLR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXCTLR_EL1	0b11	0b000	0b0101	0b0100	0b001

MSR ERXCTLR_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXCTLR_EL1	0b11	0b000	0b0101	0b0100	0b001

Accessibility

MRS <Xt>, ERXCTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXCTLR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXCTLR_EL1;
elsif PSTATE.EL == EL3 then
    return ERXCTLR_EL1;
```

MSR ERXCTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

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

B.11.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register

Accesses ext-ERR<n>STATUS for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

`0x0`

Bit descriptions

Figure B-144: AArch64_erxstatus_el1 bit assignments

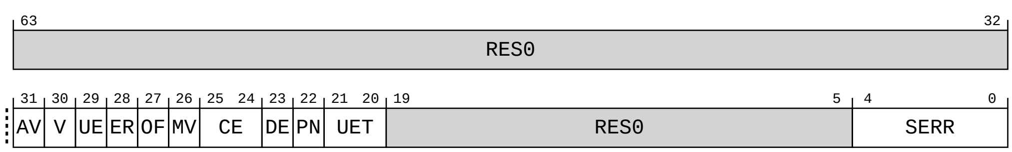


Table B-379: ERXSTATUS_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31]	AV	Address Valid. The possible values are: **`0b0`** ERXADDR_EL1 not valid. **`0b1`** ERXADDR_EL1 contains an address associated with the highest priority error recorded by this record. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	`0x0`

Bits	Name	Description	Reset
[30]	V	Status Register Valid. The possible values are: **0b0** ERXSTATUS_EL1 not valid. **0b1** ERXSTATUS_EL1 valid. At least one error has been recorded. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0x0
[29]	UE	Uncorrected Error. The possible values are: **0b0** No errors have been detected, or all detected errors have been either corrected or deferred. **0b1** At least one detected error was not corrected and not deferred. When clearing ERXSTATUS_EL1.V to 0, if this bit is nonzero, then Arm recommends that software write 1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0x0
[28]	ER	Error Reported. The possible values are: **0b0** No in-band error (External Abort) reported. **0b1** An External Abort was signaled by the node to the master making the access or other transaction. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset **Note:** An External Abort signaled by the node might be masked and not generate any exception.	0x0

Bits	Name	Description	Reset
[27]	OF	Overflow. The possible values are: **`0b0`** If UE == 1, then no error status for an Uncorrected error has been discarded. If UE == 0 and DE == 1, then no error status for a Deferred error has been discarded. If UE == 0, DE == 0, and CE !== 0b00, then the corrected error counter has not overflowed. **`0b1`** More than one error has occurred and so details of the other error have been discarded. When clearing ERXSTATUS_EL1.V to 0, if this bit is nonzero, then Arm recommends that software write 1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0. Cold reset only. Unaffected by Warm reset This bit is read/write-one-to-clear.	`0x0`
[26]	MV	Miscellaneous Registers Valid. The possible values are: **`0b0`** ERXMISC<m>_EL1 not valid. **`0b1`** This bit indicates that the ERXMISC<m>_EL1 registers contain additional information for an error recorded by this record. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset **Note:** If the ERXMISC<m>_EL1 registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded.	`0x0`

Bits	Name	Description	Reset
[25:24]	CE	Corrected Error. The possible values are: **0b00** No errors were corrected. **0b01** At least one transient error was corrected. **0b10** At least one error was corrected. **0b11** At least one persistent error was corrected. When clearing ERXSTATUS_EL1.V to 0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. This field is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0. This field is read/write-ones-to-clear. Writing a value other than all-zeros or all-ones sets this field to an UNKNOWN value. Cold reset only. Unaffected by Warm reset	0x0
[23]	DE	Deferred Error. The possible values are: **0b0** No errors were deferred. **0b1** At least one error was not corrected and deferred. When clearing ERXSTATUS_EL1.V to 0, if this bit is nonzero, then Arm recommends that software write 1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0x0

Bits	Name	Description	Reset
[22]	PN	Poison. The value is: **0b0** This core cannot distinguish a poisoned value from a corrupted value. When clearing ERXSTATUS_EL1.V to 0, if this bit is nonzero, then Arm recommends that software write 1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if any of the following are true: • ERXSTATUS_EL1.V == 0. • ERXSTATUS_EL1.{DE,UE} == {0,0}. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0x0
[21:20]	UET	Uncorrected Error Type. The value is: **0b00** Uncorrected error, Uncontainable error (UC). Cold reset only. Unaffected by Warm reset	0x0
[19:5]	RES0	Reserved	0x0
[4:0]	SERR	Primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry. The possible values are: **0b00000** No error **0b00010** ECC error from internal data buffer. **0b00110** ECC error on cache data RAM. **0b00111** ECC error on cache tag or dirty RAM. **0b01000** Parity error on TLB data RAM. **0b10010** Error response for a cache copyback. **0b10101** Deferred error from slave not supported at the consumer. For example, poisoned data received from a slave by a master that cannot defer the error further. Cold reset only. Unaffected by Warm reset	0x0

Access

MRS <Xt>, ERXSTATUS_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXSTATUS_EL1	0b11	0b000	0b0101	0b0100	0b010

MSR ERXSTATUS_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXSTATUS_EL1	0b11	0b000	0b0101	0b0100	0b010

Accessibility

MRS <Xt>, ERXSTATUS_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXSTATUS_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXSTATUS_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        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 EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXSTATUS_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXSTATUS_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXSTATUS_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXSTATUS_EL1 = X[t];
```

B.11.6 ERXADDR_EL1, Selected Error Record Address Register

Accesses ext-ERR<n>ADDR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure B-145: AArch64_erxaddr_el1 bit assignments

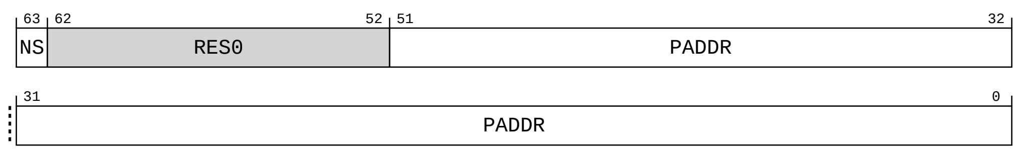


Table B-382: ERXADDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure attribute. **0b0** The address is Secure. **0b1** The address is Non-secure. Unaffected by Cold or Warm reset.	
[62:52]	RES0	Reserved	0x0
[51:0]	PADDR	Physical Address. Address of the recorded location. Unaffected by Cold or Warm reset.	

Access

MRS <Xt>, ERXADDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXADDR_EL1	0b11	0b000	0b0101	0b0100	0b011

MSR ERXADDR_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXADDR_EL1	0b11	0b000	0b0101	0b0100	0b011

Accessibility

MRS <Xt>, ERXADDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXADDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXADDR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXADDR_EL1;
elsif PSTATE.EL == EL3 then
    return ERXADDR_EL1;
```

MSR ERXADDR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXADDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXADDR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXADDR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXADDR_EL1 = X[t];
```

B.11.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register

Accesses ext-ERR<n>PFGF for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure B-146: AArch64_erxpfgf_el1 bit assignments

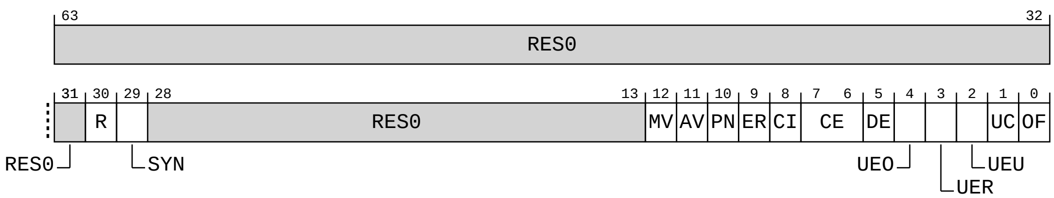


Table B-385: ERXPFGF_EL1 bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	`0x0`
[30]	R	Restartable bit. When it reaches zero, the Error Generation Counter restarts from the ERXPFGCDN_EL1 value or stops. The value is: **`0b1`** Feature controllable.	
[29]	SYN	Syndrome. Fault syndrome injection. The value is: **`0b0`** When an injected error is recorded, the node sets ERXSTATUS_EL1.{IERR, SERR} to IMPLEMENTATION DEFINED values. ERXSTATUS_EL1.{IERR, SERR} are UNKNOWN when ERXSTATUS_EL1.V == 0b0.	
[28:13]	RES0	Reserved	`0x0`
[12]	MV	Miscellaneous syndrome. Additional syndrome injection. Defines whether software can control all or part of the syndrome recorded in the ERXMISC<m>_EL1 registers when an injected error is recorded. It is **IMPLEMENTATION DEFINED** which syndrome fields in ERXMISC<m>_EL1 this refers to, as some fields might always be recorded by an error. For example, a Corrected Error counter. **`0b0`** When an injected error is recorded, the node might record IMPLEMENTATION DEFINED additional syndrome in ERXMISC<m>_EL1. If any syndrome is recorded in ERXMISC<m>_EL1, then ERXSTATUS_EL1.MV is set to 0b1. **Note:** If ERR<n>PFGF.MV == 1, software can write specific values into the ERR<n>MISC<m> registers when setting up a fault injection event. The values that can be written to these registers are **IMPLEMENTATION DEFINED**.	

Bits	Name	Description	Reset
[11]	AV	Address syndrome. Address syndrome injection. The value is: **0b0** When an injected error is recorded, the node either sets ERXADDR_EL1 and ERXSTATUS_EL1.AV for the access, or leaves these unchanged.	
[10]	PN	Poison flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.PN status flag. The value is: **0b0** When an injected error is recorded, the node sets ERXSTATUS_EL1.PN to 0.	
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.ER status flag. The value is: **0b0** When an injected error is recorded, the node sets ERXSTATUS_EL1.ER according to the architecture-defined rules for setting the ER bit.	
[8]	CI	Critical Error flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.CI status flag. The value is: **0b0** The node does not support this type of flag This behavior replaces the architecture-defined rules for setting the CI bit.	
[7:6]	CE	Corrected Error generation. The value is: **0b01** The fault generation feature of the node allows generation of a non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10. All other values are reserved.	
[5]	DE	Deferred Error generation. The value is: **0b1** The fault generation feature of the node allows generation of this type of error.	
[4]	UEO	Latent or Restartable Error generation. The value is: **0b0** The fault generation feature of the node cannot generate this type of error.	
[3]	UER	Signaled or Recoverable Error generation. The value is: **0b0** The fault generation feature of the node cannot generate this type of error.	
[2]	UEU	Unrecoverable Error generation. The value is: **0b0** The fault generation feature of the node cannot generate this type of error.	
[1]	UC	Uncontainable Error generation. The value is: **0b1** The fault generation feature of the node allows generation of this type of error.	
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.OF status flag. The value is: **0b0** When an injected error is recorded, the node sets ERXSTATUS_EL1.OF according to the architecture-defined rules for setting the OF bit.	

Access

MRS <Xt>, ERXPFGF_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXPFGF_EL1	0b11	0b000	0b0101	0b0100	0b100

Accessibility

MRS <Xt>, ERXPFGF_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXPFGF_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGF_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGF_EL1;
elsif PSTATE.EL == EL3 then
    return ERXPFGF_EL1;
```

B.11.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register

Accesses ext-ERR<n>PFGCTL for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

0x0

Bit descriptions

Figure B-147: AArch64_erxpfgctl_el1 bit assignments

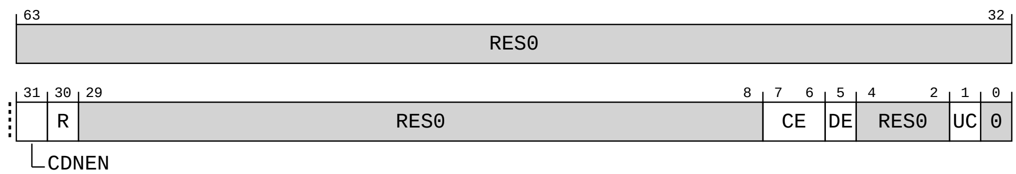


Table B-387: ERXPFGCTL_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0x0`
[31]	CDNEN	Countdown Enable. Controls transfers from the value that is held in the ERXPFGCDN_EL1 into the Error Generation Counter and enables this counter. **`0b0`** The Error Generation Counter is disabled. **`0b1`** The Error Generation Counter is enabled. On a write of 0b1 to this bit, the Error Generation Counter is set to ERXPFGCDN_EL1.CDN. Cold reset only. Unaffected by Warm reset	`0x0`
[30]	R	Restart. Controls whether, upon reaching zero, the Error Generation Counter restarts from the ERXPFGCDN_EL1 value or stops. **`0b0`** On reaching 0, the Error Generation Counter will stop. **`0b1`** On reaching 0, the Error Generation Counter is set to ERXPFGCDN_EL1.CDN. Cold reset only. Unaffected by Warm reset	`0x0`
[29:8]	RES0	Reserved	`0x0`
[7:6]	CE	Corrected Error generation enable. Controls the type of Corrected Error condition that might be generated. The possible values are: **`0b00`** No error of this type will be generated. **`0b01`** A non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10, might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	`0x0`

Bits	Name	Description	Reset
[5]	DE	Deferred Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0x0
[4:2]	RES0	Reserved	0b000
[1]	UC	Uncontainable Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0
[0]	RES0	Reserved	0b0

Access

MRS <Xt>, ERXPFGCTL_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXPFGCTL_EL1	0b11	0b000	0b0101	0b0100	0b101

MSR ERXPFGCTL_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXPFGCTL_EL1	0b11	0b000	0b0101	0b0100	0b101

Accessibility

MRS <Xt>, ERXPFGCTL_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXPFGCTL_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGCTL_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        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 EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXPFGCTL_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCTL_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCTL_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXPFGCTL_EL1 = X[t];
```

B.11.9 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

Reset value

See individual bit resets.

Bit descriptions

Figure B-148: AArch64_erxpfgcdn_el1 bit assignments

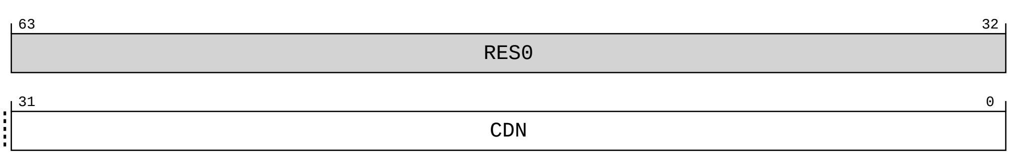


Table B-390: ERXPFGCDN_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:0]	CDN	Countdown value. This field is copied to Error Generation Counter when either: • Software writes ERXPFGCTL_EL1.CDNEN with 1. • The Error Generation Counter decrements to zero and ERXPFGCTL_EL1.R == 1. Unaffected by Cold or Warm reset. **Note:** The current Error Generation Counter value is not visible to software.	

Access

MRS <Xt>, ERXPFGCDN_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXPFGCDN_EL1	0b11	0b000	0b0101	0b0100	0b110

MSR ERXPFGCDN_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXPFGCDN_EL1	0b11	0b000	0b0101	0b0100	0b110

Accessibility

MRS <Xt>, ERXPFGCDN_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXPFGCDN_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGCDN_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGCDN_EL1;
elsif PSTATE.EL == EL3 then
    return ERXPFGCDN_EL1;
```

MSR ERXPFGCDN_EL1, <Xt>

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

```
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXPFGCDN_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXPFGCDN_EL1 = X[t];
```

B.11.10 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

Reset value

See individual bit resets.

Bit descriptions

Figure B-149: AArch64_erxmisc0_el1 bit assignments

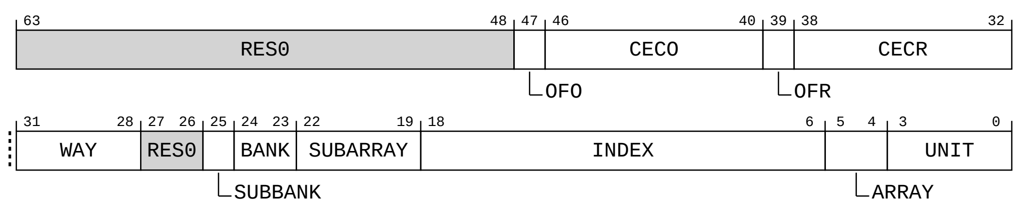


Table B-393: ERXMISC0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERXMISC0_EL1.CECO is incremented and wraps through zero. **`0b0`** Other counter has not overflowed. **`0b1`** Other counter has overflowed. A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an UNKNOWN value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an UNKNOWN value. Unaffected by Cold or Warm reset.	
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERXMISC0_EL1.CECR. Unaffected by Cold or Warm reset.	
[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERXMISC0_EL1.CECR is incremented and wraps through zero. **`0b0`** Repeat counter has not overflowed. **`0b1`** Repeat counter has overflowed. A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an UNKNOWN value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an UNKNOWN value. Unaffected by Cold or Warm reset.	
[38:32]	CECR	Corrected error count, repeat. Incremented for the first countable error, which also records other syndrome for the error, and subsequently for each countable error that matches the recorded other syndrome. This field resets to an **IMPLEMENTATION DEFINED** which might be UNKNOWN on a Cold reset. If the reset value is UNKNOWN, then the value of this field remains UNKNOWN until software initializes it. Unaffected by Cold or Warm reset.	
[31:28]	WAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L1 Data Cache] • Indicates which Tag RAM way or data RAM way detected the error. Upper 2 bits are unused. [L2 TLB] • Indicates which RAM detected an error. The possible values are 0 (RAM 1) to 9 (RAM 10). [L1 Instruction Cache] • Indicates which way detected the error. Upper 2 bits are unused. [L2 Cache] • Indicates which way detected the error. Upper 1 bit unused. Unaffected by Cold or Warm reset.	
[27:26]	RES0	Reserved	`0b00`

Bits	Name	Description	Reset
[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 Cold or Warm reset.	
[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. Unaffected by Cold or Warm 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 (0000: bank0 , 0001: bank1) Unaffected by Cold or Warm reset.	
[18:6]	INDEX	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. [L1 Data Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size [L2 TLB] • Index of TLB RAM. Upper 4 bits are unused. [L1 Instruction Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. Unaffected by Cold or Warm reset.	

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: • 00 L2 Tag RAM. • 01 L2 Data RAM. • 11 CHI Error. [L1 Data Cache] Indicates which array detected the error. The possible values are: • 00 LS Tag RAM 0. • 01 LS Tag RAM 1. • 10 LS Data RAM. • 11 LS Tag RAM 2. [L1 Instruction Cache] Indicates which array that detected the error, Data Array has higher priority. The possible values are: • 00 Tag. • 01 Data. • 10 Macro-OP cache. Unaffected by Cold or Warm reset.	
[3:0]	UNIT	Indicates the unit which detected the error. The possible values are: **0b0001** L1 Instruction Cache. **0b0010** L2 TLB. **0b0100** L1 Data Cache. **0b1000** L2 Cache.	

Access

MRS <Xt>, ERXMISC0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC0_EL1	0b11	0b000	0b0101	0b0101	0b000

MSR ERXMISC0_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC0_EL1	0b11	0b000	0b0101	0b0101	0b000

Accessibility

MRS <Xt>, ERXMISC0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC0_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        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 EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXMISC0_EL1 = X[t];
```

B.11.11 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1

Accesses ext-ERR<n>MISC1 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

0x0

Bit descriptions

Figure B-150: AArch64_erxmisc1_el1 bit assignments

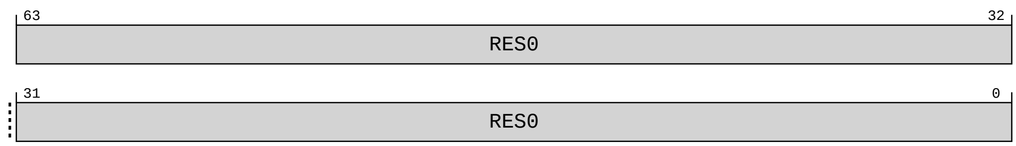


Table B-396: ERXMISC1_EL1 bit descriptions

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

Access

MRS <Xt>, ERXMISC1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC1_EL1	0b11	0b000	0b0101	0b0101	0b001

MSR ERXMISC1_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC1_EL1	0b11	0b000	0b0101	0b0101	0b001

Accessibility

MRS <Xt>, ERXMISC1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC1_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
```

```
        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 EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC1_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC1_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXMISC1_EL1 = X[t];
```

B.11.12 ERXMISC2_EL1, Selected Error Record Miscellaneous Register 2

Accesses ext-ERR<n>MISC2 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

`0x0`

Bit descriptions

Figure B-151: AArch64_erxmisc2_el1 bit assignments

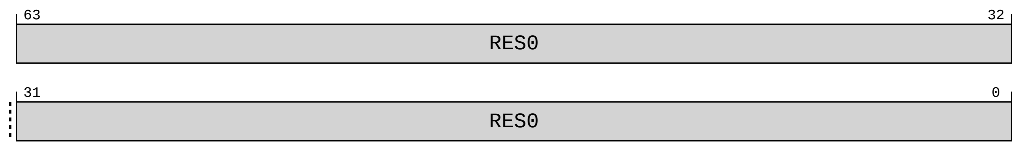


Table B-399: ERXMISC2_EL1 bit descriptions

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

Access

MRS <Xt>, ERXMISC2_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC2_EL1	0b11	0b000	0b0101	0b0101	0b010

MSR ERXMISC2_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC2_EL1	0b11	0b000	0b0101	0b0101	0b010

Accessibility

MRS <Xt>, ERXMISC2_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC2_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC2_EL1;
elsif PSTATE.EL == EL3 then
    return ERXMISC2_EL1;
```

MSR ERXMISC2_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC2_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC2_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXMISC2_EL1 = X[t];
```

B.11.13 ERXMISC3_EL1, Selected Error Record Miscellaneous Register 3

Accesses ext-ERR<n>MISC3 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

`0x0`

Bit descriptions

Figure B-152: AArch64_erxmisc3_el1 bit assignments

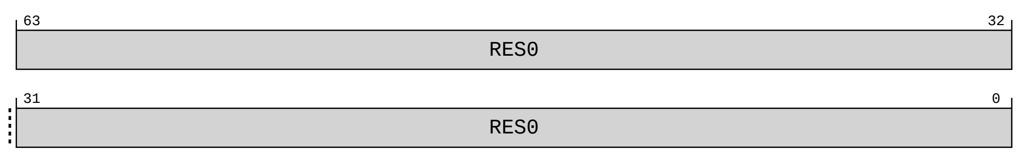


Table B-402: ERXMISC3_EL1 bit descriptions

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

Access

MRS <Xt>, ERXMISC3_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC3_EL1	0b11	0b000	0b0101	0b0101	0b011

MSR ERXMISC3_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC3_EL1	0b11	0b000	0b0101	0b0101	0b011

Accessibility

MRS <Xt>, ERXMISC3_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC3_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC3_EL1;
elsif PSTATE.EL == EL3 then
    return ERXMISC3_EL1;
```

MSR ERXMISC3_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC3_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC3_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXMISC3_EL1 = X[t];
```

B.12 AArch64 statistical profiling extension registers

The summary table provides an overview of all implementation defined statistical profiling extension registers in the core. Individual register descriptions provide detailed information.

Table B-405: statistical profiling extension register summary

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

B.12.1 PMBIDR_EL1, Profiling Buffer ID Register

Provides information to software as to whether the buffer can be programmed at the current Exception level.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

statistical-profiling-extension

Reset value

See individual bit resets.

Bit descriptions

Figure B-153: AArch64_pmbidr_el1 bit assignments

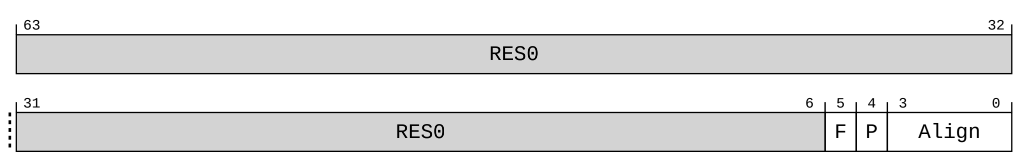


Table B-406: PMBIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:6]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[5]	F	Flag updates. Defines whether the address translation performed by the Profiling Buffer manages the Access Flag and dirty state. **0b1** Hardware management for the Access Flag and dirty state for accesses made by the Statistical Profiling Extension is controlled in the same way as explicit memory accesses in the owning translation regime.	
[4]	P	Programming not allowed. The Profiling Buffer is owned by a higher Exception level or the other Security state. **0b0** Profiling Buffer is owned by the current or a lower Exception level in the current Security state.	
[3:0]	Align	Defines the minimum alignment constraint for AArch64-PMBPTR_EL1. If this field is non-zero, then the PE must pad every record up to a multiple of this size. **0b0110** 64 Bytes.	

Access

MRS <Xt>, PMBIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
PMBIDR_EL1	0b11	0b000	0b1001	0b1010	0b111

Accessibility

MRS <Xt>, PMBIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    return PMBIDR_EL1;
elsif PSTATE.EL == EL2 then
    return PMBIDR_EL1;
elsif PSTATE.EL == EL3 then
    return PMBIDR_EL1;
```

B.12.2 PMSEVFR_EL1, Sampling Event Filter Register

Controls sample filtering by events. The overall filter is the logical AND of these filters. For example, if E[3] and E[5] are both set to 1, only samples that have both event 3 (Level 1 unified or data cache refill) and event 5 set (TLB walk) are recorded

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

statistical-profiling-extension

Reset value

See individual bit resets.

Bit descriptions

Figure B-154: AArch64_pmsevfr_el1 bit assignments

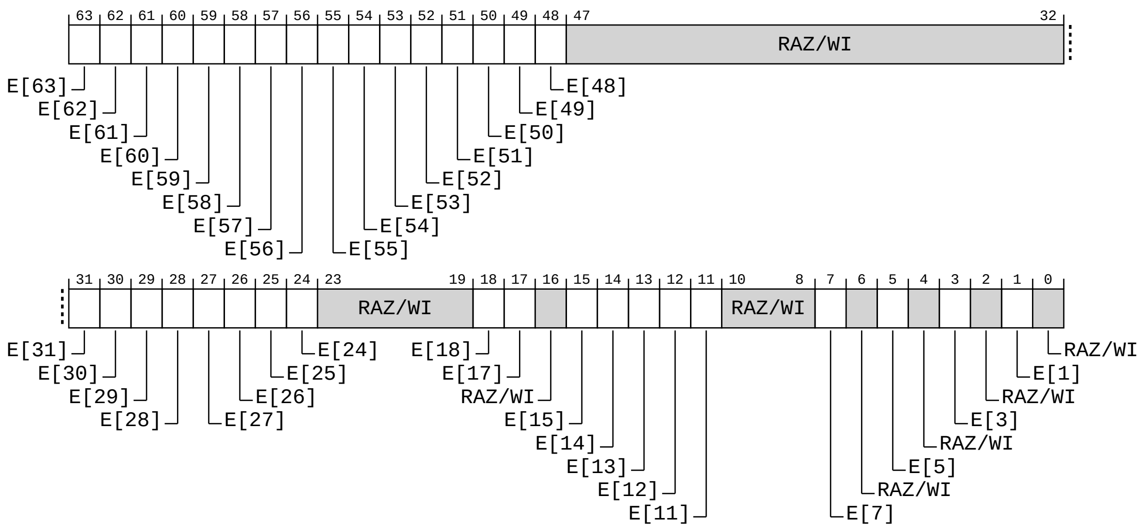


Table B-408: PMSEVFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	E[63]	E[63] is the event filter for event 63. If event 63 is not implemented, or filtering on event 63 is not supported, the corresponding bit is RAZ/WI. **`0b0`** Event 63 is ignored. **`0b1`** Do not record samples that have event 63 == 0. An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event. This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	
[62]	E[62]	E[62] is the event filter for event 62. If event 62 is not implemented, or filtering on event 62 is not supported, the corresponding bit is RAZ/WI. **`0b0`** Event 62 is ignored. **`0b1`** Do not record samples that have event 62 == 0. An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event. This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

Bits	Name	Description	Reset

[61]	E[61]	E[61] is the event filter for event 61. If event 61 is not implemented, or filtering on event 61 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 61 is ignored.
		`0b1`
		Do not record samples that have event 61 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[60]	E[60]	E[60] is the event filter for event 60. If event 60 is not implemented, or filtering on event 60 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 60 is ignored.
		`0b1`
		Do not record samples that have event 60 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[59]	E[59]	E[59] is the event filter for event 59. If event 59 is not implemented, or filtering on event 59 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 59 is ignored.
		`0b1`
		Do not record samples that have event 59 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[58]	E[58]	E[58] is the event filter for event 58. If event 58 is not implemented, or filtering on event 58 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 58 is ignored.
		`0b1`
		Do not record samples that have event 58 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

Bits	Name	Description	Reset

[57]	E[57]	E[57] is the event filter for event 57. If event 57 is not implemented, or filtering on event 57 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 57 is ignored.
		`0b1`
		Do not record samples that have event 57 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[56]	E[56]	E[56] is the event filter for event 56. If event 56 is not implemented, or filtering on event 56 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 56 is ignored.
		`0b1`
		Do not record samples that have event 56 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[55]	E[55]	E[55] is the event filter for event 55. If event 55 is not implemented, or filtering on event 55 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 55 is ignored.
		`0b1`
		Do not record samples that have event 55 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[54]	E[54]	E[54] is the event filter for event 54. If event 54 is not implemented, or filtering on event 54 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 54 is ignored.
		`0b1`
		Do not record samples that have event 54 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

Bits	Name	Description	Reset

[53]	E[53]	E[53] is the event filter for event 53. If event 53 is not implemented, or filtering on event 53 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 53 is ignored.
		`0b1`
		Do not record samples that have event 53 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[52]	E[52]	E[52] is the event filter for event 52. If event 52 is not implemented, or filtering on event 52 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 52 is ignored.
		`0b1`
		Do not record samples that have event 52 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[51]	E[51]	E[51] is the event filter for event 51. If event 51 is not implemented, or filtering on event 51 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 51 is ignored.
		`0b1`
		Do not record samples that have event 51 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[50]	E[50]	E[50] is the event filter for event 50. If event 50 is not implemented, or filtering on event 50 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 50 is ignored.
		`0b1`
		Do not record samples that have event 50 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

Bits	Name	Description	Reset
[49]	E[49]	E[49] is the event filter for event 49. If event 49 is not implemented, or filtering on event 49 is not supported, the corresponding bit is RAZ/WI. **`0b0`** Event 49 is ignored. **`0b1`** Do not record samples that have event 49 == 0. An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event. This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	
[48]	E[48]	E[48] is the event filter for event 48. If event 48 is not implemented, or filtering on event 48 is not supported, the corresponding bit is RAZ/WI. **`0b0`** Event 48 is ignored. **`0b1`** Do not record samples that have event 48 == 0. An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event. This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	
[47:32, 23:19, 10:8, 6, 4, 2, 0]	RAZ/WI	Reserved	
[31]	E[31]	E[31] is the event filter for event 31. If event 31 is not implemented, or filtering on event 31 is not supported, the corresponding bit is RAZ/WI. **`0b0`** Event 31 is ignored. **`0b1`** Do not record samples that have event 31 == 0. An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event. This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	
[30]	E[30]	E[30] is the event filter for event 30. If event 30 is not implemented, or filtering on event 30 is not supported, the corresponding bit is RAZ/WI. **`0b0`** Event 30 is ignored. **`0b1`** Do not record samples that have event 30 == 0. An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event. This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

Bits	Name	Description	Reset

[29]	E[29]	E[29] is the event filter for event 29. If event 29 is not implemented, or filtering on event 29 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 29 is ignored.
		`0b1`
		Do not record samples that have event 29 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[28]	E[28]	E[28] is the event filter for event 28. If event 28 is not implemented, or filtering on event 28 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 28 is ignored.
		`0b1`
		Do not record samples that have event 28 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[27]	E[27]	E[27] is the event filter for event 27. If event 27 is not implemented, or filtering on event 27 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 27 is ignored.
		`0b1`
		Do not record samples that have event 27 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[26]	E[26]	E[26] is the event filter for event 26. If event 26 is not implemented, or filtering on event 26 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 26 is ignored.
		`0b1`
		Do not record samples that have event 26 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

Bits	Name	Description	Reset

[25]	E[25]	E[25] is the event filter for event 25. If event 25 is not implemented, or filtering on event 25 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 25 is ignored.
		`0b1`
		Do not record samples that have event 25 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[24]	E[24]	E[24] is the event filter for event 24. If event 24 is not implemented, or filtering on event 24 is not supported, the corresponding bit is RAZ/WI.
		`0b0`
		Event 24 is ignored.
		`0b1`
		Do not record samples that have event 24 == 0.
		An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event.
		This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

[18]	E[18]	Empty predicate.
		`0b0`
		Empty predicate event is ignored.
		`0b1`
		Do not record samples that have the Empty predicate event == 0.
		This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.	

[17]	E[17]	Partial predicate.
		`0b0`
		Partial predicate event is ignored.
		`0b1`
		Do not record samples that have the Partial predicate event == 0.
		This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.	

[16]	RAZ/WI	Reserved	

Bits	Name	Description	Reset
[15]	E[15]	E[15] is the event filter for event 15. If event 15 is not implemented, or filtering on event 15 is not supported, the corresponding bit is RAZ/WI. **0b0** Event 15 is ignored. **0b1** Do not record samples that have event 15 == 0. An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event. This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	
[14]	E[14]	E[14] is the event filter for event 14. If event 14 is not implemented, or filtering on event 14 is not supported, the corresponding bit is RAZ/WI. **0b0** Event 14 is ignored. **0b1** Do not record samples that have event 14 == 0. An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event. This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	
[13]	E[13]	E[13] is the event filter for event 13. If event 13 is not implemented, or filtering on event 13 is not supported, the corresponding bit is RAZ/WI. **0b0** Event 13 is ignored. **0b1** Do not record samples that have event 13 == 0. An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event. This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	
[12]	E[12]	E[12] is the event filter for event 12. If event 12 is not implemented, or filtering on event 12 is not supported, the corresponding bit is RAZ/WI. **0b0** Event 12 is ignored. **0b1** Do not record samples that have event 12 == 0. An **IMPLEMENTATION DEFINED** event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an **IMPLEMENTATION DEFINED** filter for the event. This field is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0	

Bits	Name	Description	Reset
[11]	E[11]	Alignment. **0b0** Alignment event is ignored. **0b1** Do not record samples that have the Alignment event == 0. This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.	
[7]	E[7]	Mispredicted. **0b0** Mispredicted event is ignored. **0b1** Do not record samples that have the Mispredicted event == 0. This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.	
[5]	E[5]	TLB walk. **0b0** TLB walk event is ignored. **0b1** Do not record samples that have the TLB walk event == 0. This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.	
[3]	E[3]	Level 1 data or unified cache refill. **0b0** Level 1 data or unified cache refill event is ignored. **0b1** Do not record samples that have the Level 1 data or unified cache refill event == 0. This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0.	
[1]	E[1]	Architecturally retired. When the PE supports sampling of speculative instructions: **When the PE supports sampling of speculative instructions** **0b0** Architecturally retired event is ignored. **0b1** Do not record samples that have the Architecturally retired event == 0.	

Access

MRS <Xt>, PMSEVFR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
PMSEVFR_EL1	0b11	0b000	0b1001	0b1001	0b101

MSR PMSEVFR_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
PMSEVFR_EL1	0b11	0b000	0b1001	0b1001	0b101

Accessibility

MRS <Xt>, PMSEVFR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMSEVFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        return NVMem[0x830];
    else
        return PMSEVFR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMSEVFR_EL1;
elsif PSTATE.EL == EL3 then
    return PMSEVFR_EL1;
```

MSR PMSEVFR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMSEVFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        NVMem[0x830] = X[t];
    else
        PMSEVFR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSEVFR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    PMSEVFR_EL1 = X[t];
```

B.12.3 PMSIDR_EL1, Sampling Profiling ID Register

Describes the Statistical Profiling implementation to software

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

statistical-profiling-extension

Reset value

See individual bit resets.

Bit descriptions

Figure B-155: AArch64_pmsidr_el1 bit assignments

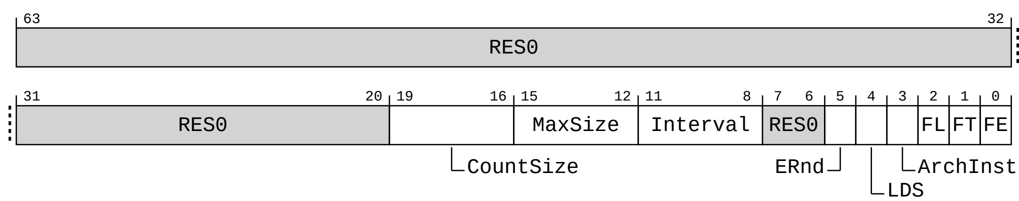


Table B-411: PMSIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[19:16]	CountSize	Defines the size of the counters `0b0010` 12-bit saturating counters	
[15:12]	MaxSize	Defines the largest size for a single record, rounded up to a power-of-two. If this is the same as the minimum alignment (PMBIDR_EL1.Align), then each record is exactly this size `0b0110` 64 bytes	
[11:8]	Interval	Recommended minimum sampling interval. This provides guidance from the implementer to the smallest minimum sampling interval, N. `0b0100` 1,024	
[5]	ERnd	Defines how the random number generator is used in determining the interval between samples, when enabled by PMSIRR_EL1.RND. `0b0` The random number is added at the start of the interval, and the sample is taken and a new interval started when the combined interval expires.	

Bits	Name	Description	Reset
[4]	LDS	Data source indicator for sampled load instructions **0b1** Loaded data source implemented	
[3]	ArchInst	Architectural instruction profiling **0b0** Micro-op sampling implemented	
[2]	FL	Filtering by latency. This bit reads as one.	
[1]	FT	Filtering by operation type. This bit reads as one.	
[0]	FE	Filtering by events. This bit reads as one.	
[63:20, 7:6]	RES0	Reserved	0b0

Access

MRS <Xt>, PMSIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
PMSIDR_EL1	0b11	0b000	0b1001	0b1001	0b111

Accessibility

MRS <Xt>, PMSIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMSIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMSIDR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.NS == '0' && MDCR_EL3.NSPB != '01' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif SCR_EL3.NS == '1' && MDCR_EL3.NSPB != '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMSIDR_EL1;
elsif PSTATE.EL == EL3 then
    return PMSIDR_EL1;
```

Appendix C ext registers

This appendix contains the descriptions for the Neoverse™ N2 ext registers.

C.1 External CoreROM register summary

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

Table C-1: External CoreROM register summary

Name	Reset	Width	Description
COREROM_ROMENTRY0	See individual bit resets.	32-bit	Core ROM table Entry 0
COREROM_ROMENTRY1	See individual bit resets.	32-bit	Core ROM table Entry 1
COREROM_ROMENTRY2	See individual bit resets.	32-bit	Core ROM table Entry 2
COREROM_ROMENTRY3	See individual bit resets.	32-bit	Core ROM table Entry 3
COREROM_AUTHSTATUS	See individual bit resets.	32-bit	Core ROM table Authentication Status Register
COREROM_DEVARCH	See individual bit resets.	32-bit	Core ROM table Device Architecture Register
COREROM_DEVTYPE	See individual bit resets.	32-bit	Core ROM table Device Type Register
COREROM_PIDR4	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 4
COREROM_PIDR0	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 0
COREROM_PIDR1	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 1
COREROM_PIDR2	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 2
COREROM_PIDR3	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 3
COREROM_CIDR0	See individual bit resets.	32-bit	Core ROM table Component Identification Register 0
COREROM_CIDR1	See individual bit resets.	32-bit	Core ROM table Component Identification Register 1
COREROM_CIDR2	See individual bit resets.	32-bit	Core ROM table Component Identification Register 2
COREROM_CIDR3	See individual bit resets.	32-bit	Core ROM table Component Identification Register 3

C.1.1 COREROM_ROMENTRY0, Core ROM table Entry 0

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0x000

Reset value

See individual bit resets.

Bit descriptions

Figure C-1: ext_corerom_romentry0 bit assignments

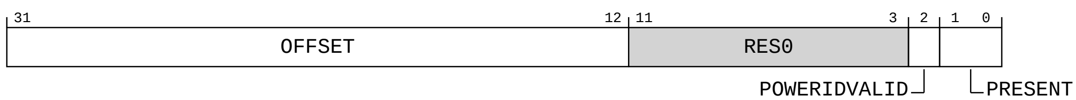


Table C-2: COREROM_ROMENTRY0 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). **`0b00000000000000010000`** Core DBG component at address 0x1_0000.	
[11:3]	RES0	Reserved	`0b000000000`
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **`0b0`** A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **`0b11`** The ROM Entry is present.	

C.1.2 COREROM_ROMENTRY1, Core ROM table Entry 1

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0x004

Reset value

See individual bit resets.

Bit descriptions

Figure C-2: ext_corerom_romentry1 bit assignments

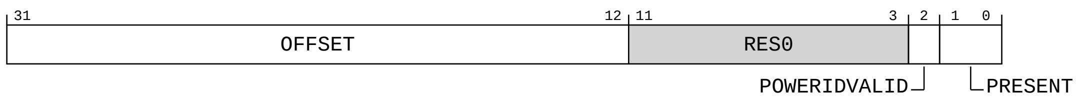


Table C-3: COREROM_ROMENTRY1 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). **`0b00000000000000100000`** CORE PMU component at address 0x2_0000.	
[11:3]	RES0	Reserved	`0b000000000`
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **`0b0`** A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **`0b11`** The ROM Entry is present.	

C.1.3 COREROM_ROMENTRY2, Core ROM table Entry 2

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0x008

Reset value

See individual bit resets.

Bit descriptions

Figure C-3: ext_corerom_romentry2 bit assignments

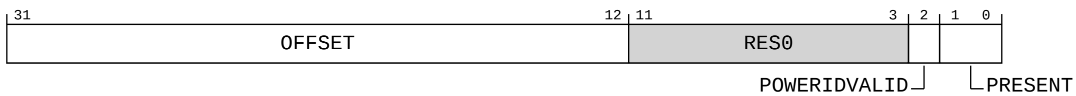


Table C-4: COREROM_ROMENTRY2 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). **`0b00000000000000110000`** Core ETM component at address 0x3_0000.	
[11:3]	RES0	Reserved	`0b000000000`
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **`0b0`** A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **`0b11`** The ROM Entry is present.	

C.1.4 COREROM_ROMENTRY3, Core ROM table Entry 3

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0x00C

Reset value

See individual bit resets.

Bit descriptions

Figure C-4: ext_corerom_romentry3 bit assignments

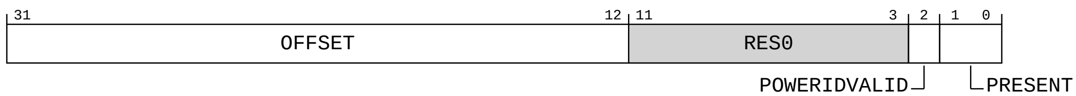


Table C-5: COREROM_ROMENTRY3 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation: Component Address = ROM Table Base Address + (OFFSET << 12). **`0b00000000000001000000`** Core ELA component at address 0x4_0000.	
[11:3]	RES0	Reserved	`0b000000000`
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **`0b0`** A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **`0b11`** The ROM Entry is present.	

C.1.5 COREROM_AUTHSTATUS, Core ROM table Authentication Status Register

Provides information about the state of the authentication interface for debug.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFB8

Reset value

See individual bit resets.

Bit descriptions

Figure C-5: ext_corerom_authstatus bit assignments

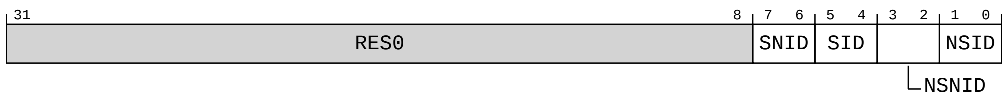


Table C-6: COREROM_AUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0x0
[7:6]	SNID	Secure Non-invasive Debug. ExternalSecureNoninvasiveDebugEnabled() == ExternalSecureInvasiveDebugEnabled(). This field has the same value as the SID field.	
[5:4]	SID	Secure Invasive Debug. **`0b10`** Secure invasive debug disabled. ExternalSecureInvasiveDebugEnabled() == FALSE. **`0b11`** Secure invasive debug enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.	
[3:2]	NSNID	Non-secure Non-invasive Debug. **`0b00`** Debug level is not supported.	
[1:0]	NSID	Non-secure Invasive Debug. **`0b00`** Debug level is not supported.	

C.1.6 COREROM_DEVARCH, Core ROM table Device Architecture Register

Identifies the architect and architecture of a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFBC

Reset value

See individual bit resets.

Bit descriptions

Figure C-6: ext_corerom_devarch bit assignments

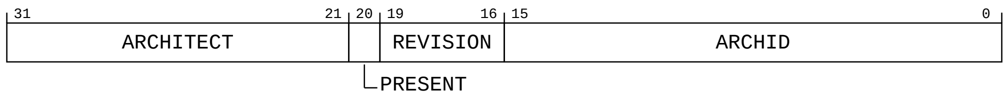


Table C-7: COREROM_DEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. **0b01000111011** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	
[20]	PRESENT	Present. **0b1** DEVARCH information present.	
[19:16]	REVISION	Revision. **0b0000** Revision 0.	
[15:0]	ARCHID	Architecture ID. **0b0000101011110111** ROM Table v0. The debug tool must inspect ext-COREROM_DEVTYPE and ext-COREROM_DEVID to determine further information about the ROM Table.	

C.1.7 COREROM_DEVTYPE, Core ROM table Device Type Register

A debugger can use DEVTYPE to obtain information about a component that has an unrecognized part number.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFCC

Reset value

See individual bit resets.

Bit descriptions

Figure C-7: ext_corerom_devtype bit assignments

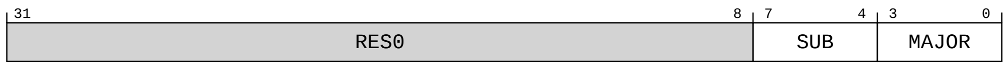


Table C-8: COREROM_DEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	SUB	Sub number **`0b0000`** Other, undefined.	
[3:0]	MAJOR	Major number **`0b0000`** Miscellaneous.	

C.1.8 COREROM_PIDR4, Core ROM table Peripheral Identification Register 4

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFD0

Reset value

See individual bit resets.

Bit descriptions

Figure C-8: ext_corerom_pidr4 bit assignments

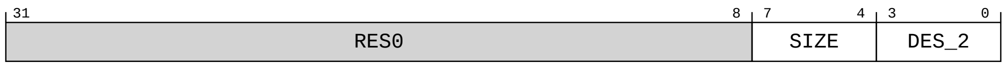


Table C-9: COREROM_PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	SIZE	4KB count. **`0b0000`** The component uses a single 4KB block.	
[3:0]	DES_2	JEP106 continuation code. **`0b0100`** Arm Limited. Number of 0x7F bytes in full JEP106 code 0x7F 0x7F 0x7F 0x7F 0x3B.	

C.1.9 COREROM_PIDR0, Core ROM table Peripheral Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFE0

Reset value

See individual bit resets.

Bit descriptions

Figure C-9: ext_corerom_pidr0 bit assignments

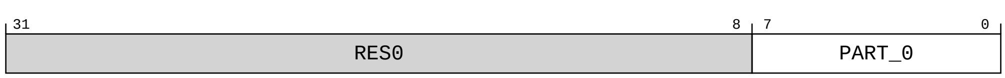


Table C-10: COREROM_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PART_0	Part number bits [7:0]. **`0b01001001`** Neoverse N2 Core ROM table. Bits [7:0] of part number 0xD49.	

C.1.10 COREROM_PIDR1, Core ROM table Peripheral Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFE4

Reset value

See individual bit resets.

Bit descriptions

Figure C-10: ext_corerom_pidr1 bit assignments

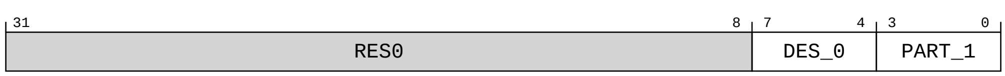


Table C-11: COREROM_PIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	DES_0	JEP106 identification code bits [3:0]. **`0b1011`** Arm Limited. Bits [3:0] of JEP106 identification code 0x3B.	
[3:0]	PART_1	Part number bits [11:8]. **`0b1101`** Neoverse N2 Core ROM table. Bits [11:8] of part number 0xD49.	

C.1.11 COREROM_PIDR2, Core ROM table Peripheral Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFE8

Reset value

See individual bit resets.

Bit descriptions

Figure C-11: ext_corerom_pidr2 bit assignments

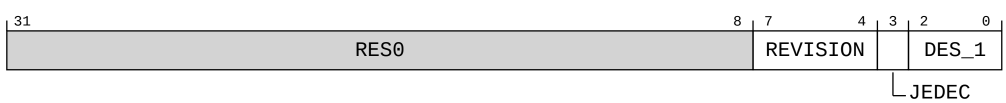


Table C-12: COREROM_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	REVISION	Part major revision **`0b0000`** Revision r0p0.	
[3]	JEDEC	JEDEC assignee. **`0b1`** JEDEC-assignee values is used.	
[2:0]	DES_1	JEP106 identification code bits [6:4]. **`0b011`** Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	

C.1.12 COREROM_PIDR3, Core ROM table Peripheral Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFEC

Reset value

See individual bit resets.

Bit descriptions

Figure C-12: ext_corerom_pidr3 bit assignments

31 … 8	7 … 4	3 … 0
RES0	REVAND	CMOD

Table C-13: COREROM_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	REVAND	Part minor revision **`0b0000`**	
[3:0]	CMOD	Customer Modified. **`0b0000`** The component is not modified from the original design.	

C.1.13 COREROM_CIDR0, Core ROM table Component Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFF0

Reset value

See individual bit resets.

Bit descriptions

Figure C-13: ext_corerom_cidr0 bit assignments

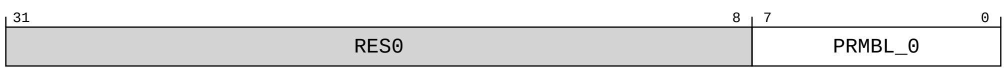


Table C-14: COREROM_CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0x0
[7:0]	PRMBL_0	CoreSight component identification preamble. **0b00001101** CoreSight component identification preamble.	

C.1.14 COREROM_CIDR1, Core ROM table Component Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFF4

Reset value

See individual bit resets.

Bit descriptions

Figure C-14: ext_corerom_cidr1 bit assignments

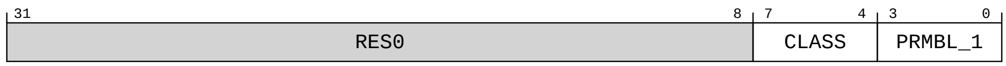


Table C-15: COREROM_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	CLASS	CoreSight component class. **`0b1001`** CoreSight component.	
[3:0]	PRMBL_1	CoreSight component identification preamble. **`0b0000`** CoreSight component identification preamble.	

C.1.15 COREROM_CIDR2, Core ROM table Component Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFF8

Reset value

See individual bit resets.

Bit descriptions

Figure C-15: ext_corerom_cidr2 bit assignments

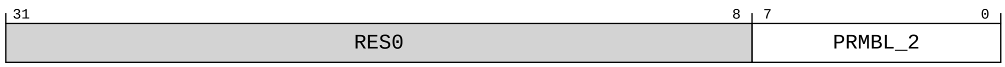


Table C-16: COREROM_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PRMBL_2	CoreSight component identification preamble. **`0b00000101`** CoreSight component identification preamble.	

C.1.16 COREROM_CIDR3, Core ROM table Component Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CoreROM

Register offset

0xFFC

Reset value

See individual bit resets.

Bit descriptions

Figure C-16: ext_corerom_cidr3 bit assignments

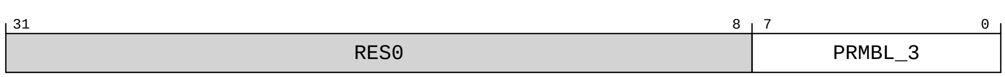


Table C-17: COREROM_CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`

Bits	Name	Description	Reset
[7:0]	PRMBL_3	CoreSight component identification preamble. **0b10110001** CoreSight component identification preamble.	

C.2 External PPM register summary

The summary table provides an overview of all external PPM registers in the core. Individual register descriptions provide detailed information.

Table C-18: External PPM register summary

Offset	Name	Reset	Width	Description
0x000	CPUPPMCR	See individual bit resets.	64-bit	Power Performance Management Register
0x010	CPUPPMCR2	See individual bit resets.	64-bit	Power Performance Management Register
0x020	CPUPPMCR3	See individual bit resets.	64-bit	Power Performance Management Register
0x080	CPUPPMCR4	See individual bit resets.	64-bit	Power Performance Management Register
0x088	CPUPPMCR5	See individual bit resets.	64-bit	Power Performance Management Register
0x090	CPUPPMCR6	See individual bit resets.	64-bit	Power Performance Management Register

C.2.1 CPUPPMCR, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x000

Reset value

See individual bit resets.

Bit descriptions

Figure C-17: ext_cpuppmcr bit assignments

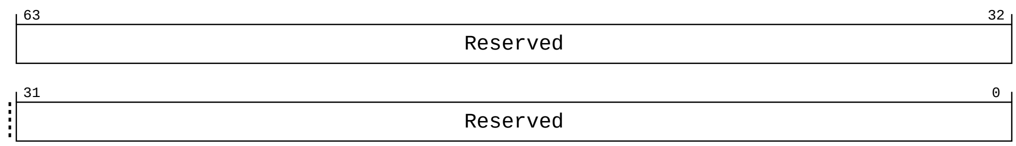


Table C-19: CPUPPMCR bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	

C.2.2 CPUPPMCR2, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x010

Reset value

See individual bit resets.

Bit descriptions

Figure C-18: ext_cpuppmcr2 bit assignments

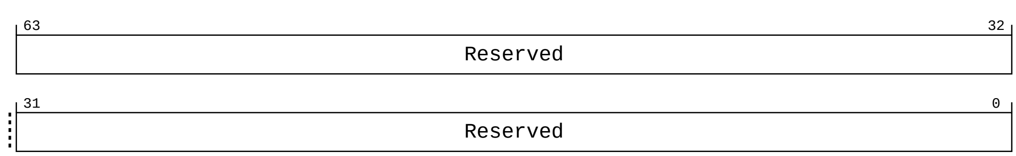


Table C-20: CPUPPMCR2 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	

C.2.3 CPUPPMCR3, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x020

Reset value

See individual bit resets.

Bit descriptions

Figure C-19: ext_cpuppmcr3 bit assignments

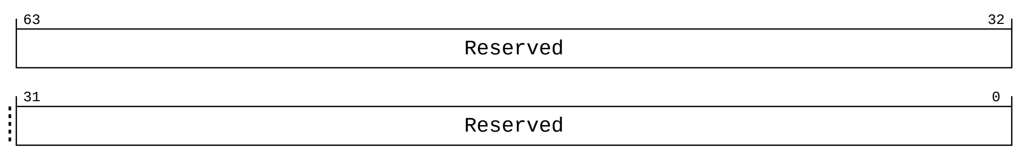


Table C-21: CPUPPMCR3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	

C.2.4 CPUPPMCR4, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x080

Reset value

See individual bit resets.

Bit descriptions

Figure C-20: ext_cpuppmcr4 bit assignments

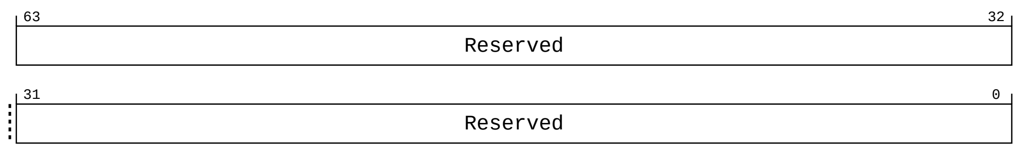


Table C-22: CPUPPMCR4 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	

C.2.5 CPUPPMCR5, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x088

Reset value

See individual bit resets.

Bit descriptions

Figure C-21: ext_cpuppmcr5 bit assignments

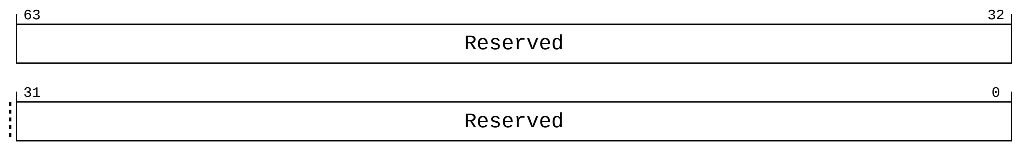


Table C-23: CPUPPMCR5 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	

C.2.6 CPUPPMCR6, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x090

Reset value

See individual bit resets.

Bit descriptions

Figure C-22: ext_cpuppmcr6 bit assignments

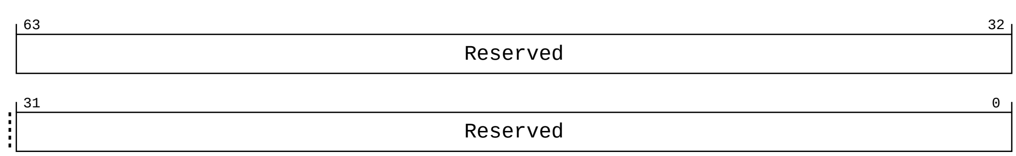


Table C-24: CPUPPMCR6 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	

C.3 External Performance monitors register summary

The summary table provides an overview of all External Performance monitors registers in the core. Individual register descriptions provide detailed information.

Table C-25: External Performance monitors register summary

Offset	Name	Reset	Width	Description
0x600	PMPCSSR	See individual bit resets.	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x60C	PMCID2SSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register
0x610	PMSSSR	`0x1`	32-bit	PMU Snapshot Status Register
0x618	PMCCNTSR	See individual bit resets.	64-bit	PMU Cycle Counter Snapshot Register
0x620	PMEVCNTSR0	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x628	PMEVCNTSR1	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x630	PMEVCNTSR2	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x638	PMEVCNTSR3	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x640	PMEVCNTSR4	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x648	PMEVCNTSR5	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F0	PMSSCR	See individual bit resets.	32-bit	PMU Snapshot Capture Register
0xE00	PMCFGR	See individual bit resets.	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_EL0	See individual bit resets.	32-bit	Performance Monitors Control Register
0xE20	PMCEID0	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 0
0xE24	PMCEID1	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 1
0xE28	PMCEID2	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 2
0xE2C	PMCEID3	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 3
0xE40	PMMIR	See individual bit resets.	32-bit	Performance Monitors Machine Identification Register
0xFBC	PMDEVARCH	See individual bit resets.	32-bit	Performance Monitors Device Architecture register
0xFC8	PMDEVID	See individual bit resets.	32-bit	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	See individual bit resets.	32-bit	Performance Monitors Device Type register
0xFD0	PMPIDR4	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 3

C.3.1 PMPCSSR, Snapshot Program Counter Sample Register

Captured copy of the Program Counter.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x600

Reset value

See individual bit resets.

Bit descriptions

Figure C-23: ext_pmpcssr bit assignments

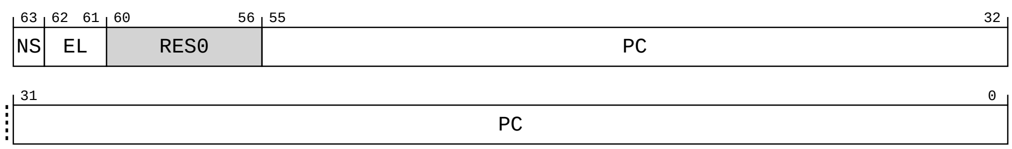


Table C-26: PMPCSSR bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure sample. **`0b0`** The captured instruction was executed in Secure state. **`0b1`** The captured instruction was executed in Non-secure state.	
[62:61]	EL	Exception level sample. The Exception level the captured instruction was executed at.	
[60:56]	RES0	Reserved	`0b00000`

Bits	Name	Description	Reset
[55:0]	PC	Sampled PC. The instruction address for the sampled instruction. The sampled instruction must be an instruction recently executed by the PE. The architecture does not require that all instructions are eligible for sampling. However, it must be possible to reference instructions at branch targets. The branch target for a conditional branch instruction that fails its Condition code check is the instruction following the conditional branch target. The sampled instruction must be architecturally executed. However, in exceptional circumstances, such as a change in security state or other boundary condition, it is permissible to sample an instruction that was speculatively executed and not architecturally executed. **Note:** The ARM architecture does not define recently executed.	

C.3.2 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register

Captured copy of the CONTEXTIDR_EL1 register.

The value captured must relate to the instruction captured in PMPCSSR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x608

Reset value

See individual bit resets.

Bit descriptions

Figure C-24: ext_pmcidssr bit assignments

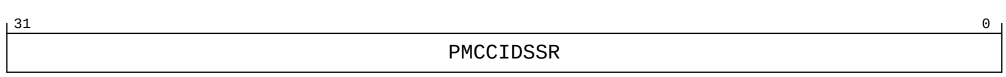


Table C-27: PMCIDSSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMCCIDSSR	PMCIDSR sample. Sampled CONTEXTIDR_EL1 snapshot.	

C.3.3 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register

Captured copy of the CONTEXTIDR_EL2 register.

The value captured must relate to the instruction captured in PMPCSSR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x60C

Reset value

See individual bit resets.

Bit descriptions

Figure C-25: ext_pmcid2ssr bit assignments

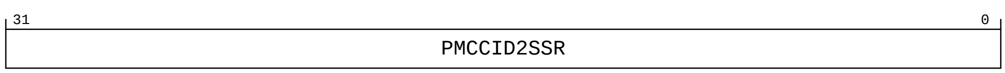


Table C-28: PMCID2SSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMCCID2SSR	PMCID2SR sample. Sampled CONTEXTIDR_EL2 snapshot.	

C.3.4 PMSSSR, PMU Snapshot Status Register

Holds status information about the captured counters.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x610

Reset value

`0x1`

Bit descriptions

Figure C-26: ext_pmsssr bit assignments

Table C-29: PMSSSR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	`0x0`
[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.	`0x1`

C.3.5 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

Reset value

See individual bit resets.

Bit descriptions

Figure C-27: ext_pmccntsr bit assignments

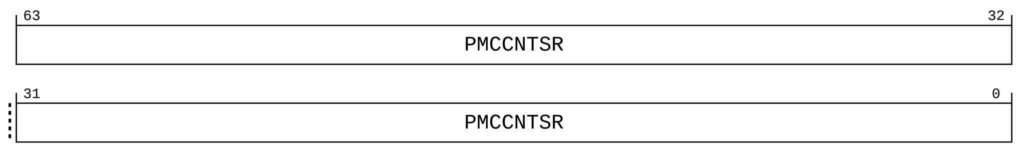


Table C-30: PMCCNTSR bit descriptions

Bits	Name	Description	Reset
[63:0]	PMCCNTSR	PMCCNTR_EL0 sample. Sampled cycle count.	

C.3.6 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

Reset value

See individual bit resets.

Bit descriptions

Figure C-28: ext_pmevcntsr0 bit assignments

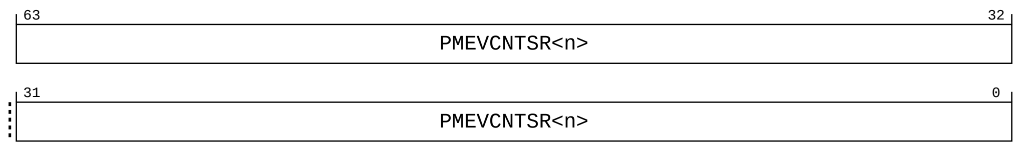


Table C-31: PMEVCNTSR0 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.3.7 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

Reset value

See individual bit resets.

Bit descriptions

Figure C-29: ext_pmevcntsr1 bit assignments

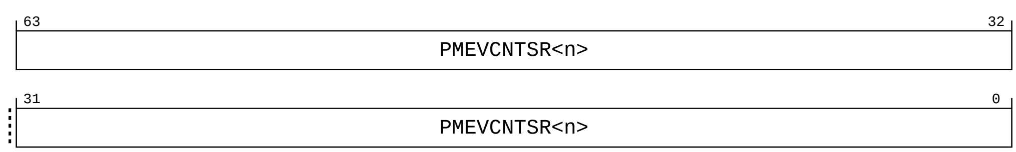


Table C-32: PMEVCNTSR1 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.3.8 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

Reset value

See individual bit resets.

Bit descriptions

Figure C-30: ext_pmevcntsr2 bit assignments

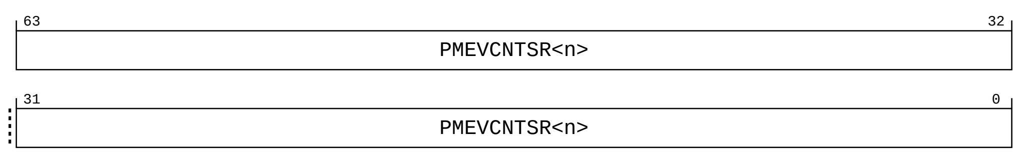


Table C-33: PMEVCNTSR2 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.3.9 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

Reset value

See individual bit resets.

Bit descriptions

Figure C-31: ext_pmevcntsr3 bit assignments

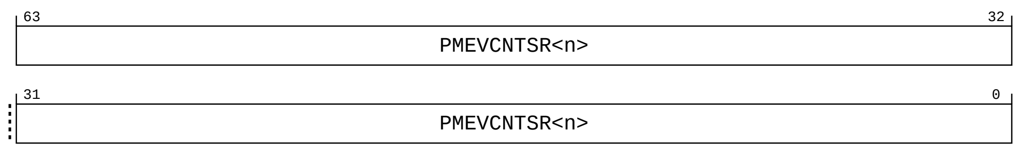


Table C-34: PMEVCNTSR3 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.3.10 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

Reset value

See individual bit resets.

Bit descriptions

Figure C-32: ext_pmevcntsr4 bit assignments

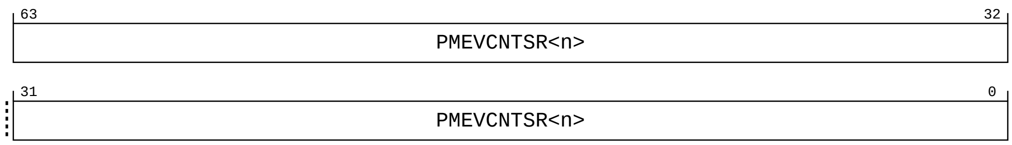


Table C-35: PMEVCNTSR4 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.3.11 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

Reset value

See individual bit resets.

Bit descriptions

Figure C-33: ext_pmevcntsr5 bit assignments

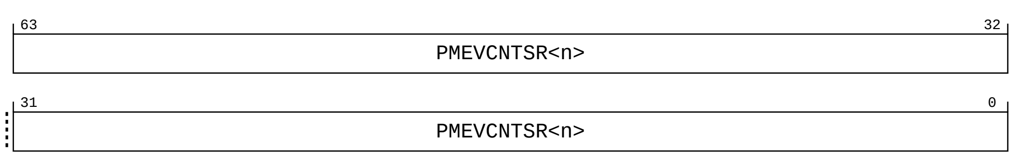


Table C-36: PMEVCNTSR5 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.3.12 PMSSCR, PMU Snapshot Capture Register

Provides a mechanism for software to initiate a sample.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x6F0

Reset value

See individual bit resets.

Bit descriptions

Figure C-34: ext_pmsscr bit assignments

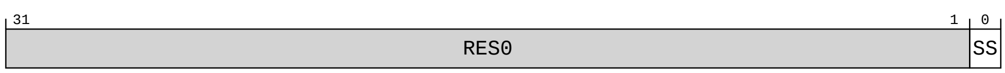


Table C-37: PMSSCR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	`0x0`
[0]	SS	Capture now. **`0b0`** Ignored. **`0b1`** Initiate a capture immediately.	

C.3.13 PMCFGR, Performance Monitors Configuration Register

Contains PMU-specific configuration data.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE00

Reset value

See individual bit resets.

Bit descriptions

Figure C-35: ext_pmcfgr bit assignments

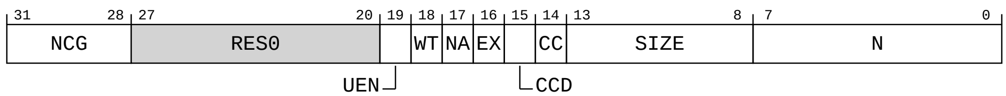


Table C-38: PMCFGR bit descriptions

Bits	Name	Description	Reset
[31:28]	NCG	This feature is not supported, so this field is RAZ.	
[27:20]	RES0	Reserved	0b00000000
[19]	UEN	User-mode Enable Register supported. AArch64-PMUSERENR_EL0 is not visible in the external debug interface, so this bit is RAZ.	
[18]	WT	This feature is not supported, so this bit is RAZ.	
[17]	NA	This feature is not supported, so this bit is RAZ.	
[16]	EX	Export supported. **0b1** ext-PMCR_EL0.X is read/write.	
[15]	CCD	Cycle counter has prescale. **0b1** ext-PMCR_EL0.D is read/write.	
[14]	CC	Dedicated cycle counter (counter 31) supported. This bit is RAO.	
[13:8]	SIZE	Size of counters, minus one. This field defines the size of the largest counter implemented by the Performance Monitors Unit. From Armv8, the largest counter is 64-bits, so the value of this field is 111111. 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.	

Bits	Name	Description	Reset
[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. Must be configured to either `0x06` or `0x14`	

C.3.14 PMCR_EL0, Performance Monitors Control Register

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configurations

This register is only partially mapped to the internal AArch32-PMCR System register. An external agent must use other means to discover the information held in AArch32-PMCR[31:11], such as accessing ext-PMCFGR and the ID registers.

Attributes

Width

32

Component

PMU

Register offset

0xE04

Reset value

See individual bit resets.

Bit descriptions

Figure C-36: ext_pmcr_el0 bit assignments

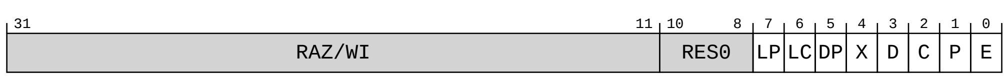


Table C-39: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[31:11]	RAZ/ WI	Reserved	
[10:8]	RES0	Reserved	`0b000`

Bits	Name	Description	Reset
[7]	LP	Long event counter enable. Determines when unsigned overflow is recorded by a counter overflow bit. **0b1** Event counter overflow on increment that causes unsigned overflow of ext-PMEVCNTR<n>_EL0[63:0].	
[6]	LC	Long cycle counter enable. Determines when unsigned overflow is recorded by the cycle counter overflow bit. **0b0** Cycle counter overflow on increment that causes unsigned overflow of ext-PMCCNTR_EL0[31:0]. **0b1** Cycle counter overflow on increment that causes unsigned overflow of ext-PMCCNTR_EL0[63:0].	
[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.	
[4]	X	Enable export of events in an **IMPLEMENTATION DEFINED** PMU event export bus. **0b0** Do not export events. **0b1** Export events where not prohibited. This field enables the exporting of events over an **IMPLEMENTATION DEFINED** PMU event export bus to another device, for example to an OPTIONAL PE trace unit. No events are exported when counting is prohibited. This field does not affect the generation of Performance Monitors overflow interrupt requests or signaling to a cross-trigger interface (CTI) that can be implemented as signals exported from the PE.	
[3]	D	Clock divider. **0b0** When enabled, ext-PMCCNTR_EL0 counts every clock cycle. **0b1** When enabled, ext-PMCCNTR_EL0 counts once every 64 clock cycles.	
[2]	C	Cycle counter reset. The effects of writing to this bit are: **0b1** Reset ext-PMCCNTR_EL0 to zero.	
[1]	P	Event counter reset. The effects of writing to this bit are: **0b1** Reset all event counters, not including ext-PMCCNTR_EL0, to zero.	
[0]	E	Enable **0b1** All event counters in the range [0..(PMN-1)] and ext-PMCCNTR_EL0, are enabled by ext-PMCNTENSET_EL0.	

C.3.15 PMCEID0, Performance Monitors Common Event Identification register 0

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x0000 to 0x001F

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'. - Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. - This view of the register was previously called PMCEID0_EL0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE20

Reset value

See individual bit resets.

Bit descriptions

Figure C-37: ext_pmceid0 bit assignments

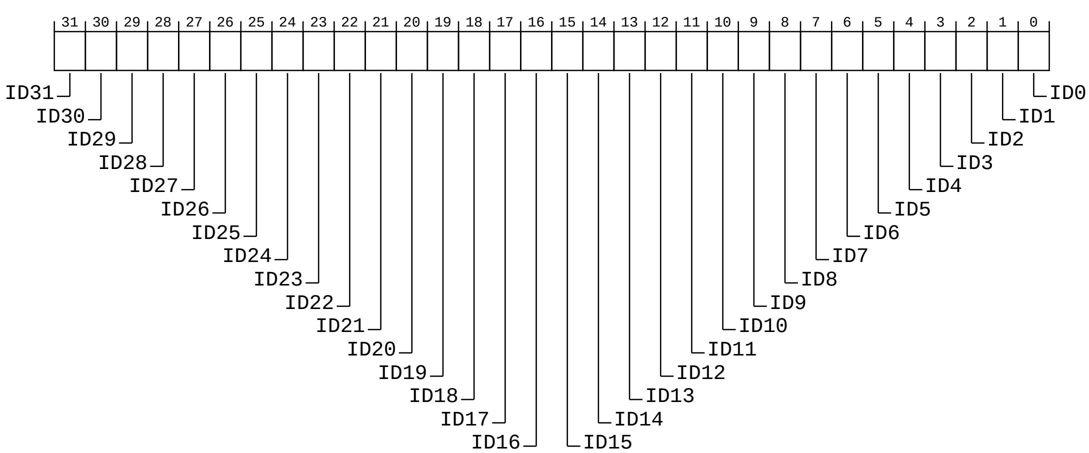


Table C-40: PMCEID0 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x1f) L1D_CACHE_ALLOCATE `0b0` The common event is not implemented, or not counted.	
[30]	ID30	ID30 corresponds to common event (0x1e) CHAIN `0b1` The common event is implemented.	
[29]	ID29	ID29 corresponds to common event (0x1d) BUS_CYCLES `0b1` The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED `0b1` The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x1b) INST_SPEC `0b1` The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x1a) MEMORY_ERROR `0b1` The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (`0x19`) BUS_ACCESS `0b1` The common event is implemented.	
[24]	ID24	ID24 corresponds to common event (`0x18`) L2D_CACHE_WB `0b1` The common event is implemented.	
[23]	ID23	ID23 corresponds to common event (`0x17`) L2D_CACHE_REFILL `0b1` The common event is implemented.	
[22]	ID22	ID22 corresponds to common event (`0x16`) L2D_CACHE `0b1` The common event is implemented.	
[21]	ID21	ID21 corresponds to common event (`0x15`) L1D_CACHE_WB `0b1` The common event is implemented.	
[20]	ID20	ID20 corresponds to common event (`0x14`) L1I_CACHE `0b1` The common event is implemented.	
[19]	ID19	ID19 corresponds to common event (`0x13`) MEM_ACCESS `0b1` The common event is implemented.	

Bits	Name	Description	Reset
[18]	ID18	ID18 corresponds to common event (`0x12`) BR_PRED **`0b1`** The common event is implemented.	
[17]	ID17	ID17 corresponds to common event (`0x11`) CPU_CYCLES **`0b1`** The common event is implemented.	
[16]	ID16	ID16 corresponds to common event (`0x10`) BR_MIS_PRED **`0b1`** The common event is implemented.	
[15]	ID15	ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[14]	ID14	ID14 corresponds to common event (0xe) BR_RETURN_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[12]	ID12	ID12 corresponds to common event (0xc) PC_WRITE_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[11]	ID11	ID11 corresponds to common event (0xb) CID_WRITE_RETIRED **`0b1`** The common event is implemented.	
[10]	ID10	ID10 corresponds to common event (0xa) EXC_RETURN **`0b1`** The common event is implemented.	
[9]	ID9	ID9 corresponds to common event (`0x9`) EXC_TAKEN **`0b1`** The common event is implemented.	
[8]	ID8	ID8 corresponds to common event (`0x8`) INST_RETIRED **`0b1`** The common event is implemented.	
[7]	ID7	ID7 corresponds to common event (`0x7`) ST_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[6]	ID6	ID6 corresponds to common event (`0x6`) LD_RETIRED **`0b0`** The common event is not implemented, or not counted.	
[5]	ID5	ID5 corresponds to common event (`0x5`) L1D_TLB_REFILL **`0b1`** The common event is implemented.	

Bits	Name	Description	Reset
[4]	ID4	ID4 corresponds to common event (`0x4`) L1D_CACHE **`0b1`** The common event is implemented.	
[3]	ID3	ID3 corresponds to common event (`0x3`) L1D_CACHE_REFILL **`0b1`** The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (`0x2`) L1I_TLB_REFILL **`0b1`** The common event is implemented.	
[1]	ID1	ID1 corresponds to common event (`0x1`) L1I_CACHE_REFILL **`0b1`** The common event is implemented.	
[0]	ID0	ID0 corresponds to common event (`0x0`) SW_INCR **`0b1`** The common event is implemented.	

C.3.16 PMCEID1, Performance Monitors Common Event Identification register 1

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x020 to 0x03F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'. - Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. - This view of the register was previously called PMCEID1_EL0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE24

Reset value

See individual bit resets.

Bit descriptions

Figure C-38: ext_pmceid1 bit assignments

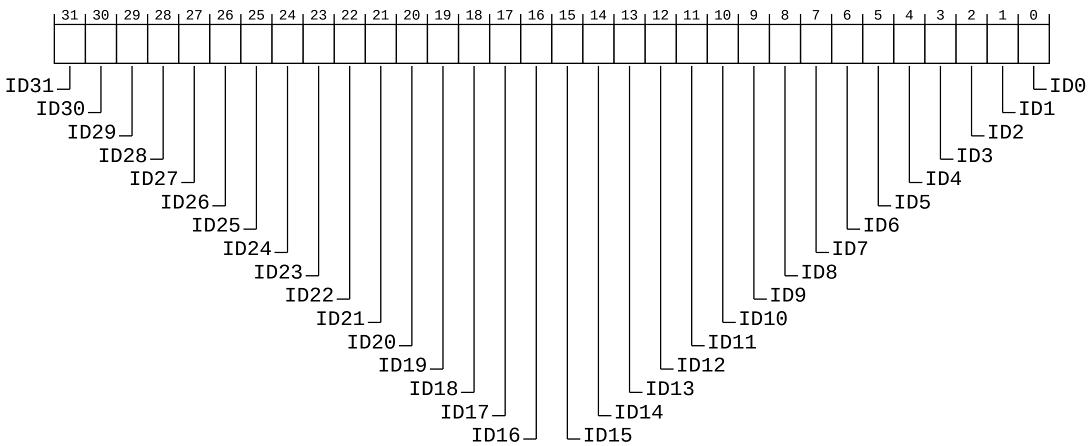


Table C-41: PMCEID1 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x3f) STALL_SLOT **0b1** The common event is implemented.	
[30]	ID30	ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND **0b1** The common event is implemented.	
[29]	ID29	ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND **0b1** The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x3c) STALL **0b1** The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x3b) OP_SPEC **0b1** The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x3a) OP_RETIRED **0b1** The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (0x39) L1D_CACHE_LMISS_RD **0b1** The common event is implemented.	
[24]	ID24	ID24 corresponds to common event (0x38) REMOTE_ACCESS_RD **0b0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[23]	ID23	ID23 corresponds to common event (`0x37`) LL_CACHE_MISS_RD **`0b1`** The common event is implemented.	
[22]	ID22	ID22 corresponds to common event (`0x36`) LL_CACHE_RD **`0b1`** The common event is implemented.	
[21]	ID21	ID21 corresponds to common event (`0x35`) ITLB_WLK **`0b1`** The common event is implemented.	
[20]	ID20	ID20 corresponds to common event (`0x34`) DTLB_WLK **`0b1`** The common event is implemented.	
[19]	ID19	ID19 corresponds to a Reserved Event event (`0x33`) **`0b0`** The common event is not implemented, or not counted.	
[18]	ID18	ID18 corresponds to a Reserved Event event (`0x32`) **`0b0`** The common event is not implemented, or not counted.	
[17]	ID17	ID17 corresponds to common event (`0x31`) REMOTE_ACCESS **`0b1`** The common event is implemented.	
[16]	ID16	ID16 corresponds to common event (`0x30`) L2I_TLB **`0b0`** The common event is not implemented, or not counted.	
[15]	ID15	ID15 corresponds to common event (0x2f) L2TLB_REQ **`0b1`** The common event is implemented.	
[14]	ID14	ID14 corresponds to common event (0x2e) L2I_TLB_REFILL **`0b0`** The common event is not implemented, or not counted.	
[13]	ID13	ID13 corresponds to common event (0x2d) L2TLB_REFILL **`0b1`** The common event is implemented.	
[12]	ID12	ID12 corresponds to common event (0x2c) Reserved **`0b0`** The common event is not implemented, or not counted.	
[11]	ID11	ID11 corresponds to common event (0x2b) L3D_CACHE **`0b1`** The common event is implemented.	
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL **`0b1`** The common event is implemented.	

Bits	Name	Description	Reset
[9]	ID9	ID9 corresponds to common event (`0x29`) L3D_CACHE_ALLOCATE **`0b1`** The common event is implemented.	
[8]	ID8	ID8 corresponds to common event (`0x28`) L2I_CACHE_REFILL **`0b0`** The common event is not implemented, or not counted.	
[7]	ID7	ID7 corresponds to common event (`0x27`) L2I_CACHE **`0b0`** The common event is not implemented, or not counted.	
[6]	ID6	ID6 corresponds to common event (`0x26`) L1I_TLB **`0b1`** The common event is implemented.	
[5]	ID5	ID5 corresponds to common event (`0x25`) L1D_TLB **`0b1`** The common event is implemented.	
[4]	ID4	ID4 corresponds to common event (`0x24`) STALL_BACKEND **`0b1`** The common event is implemented.	
[3]	ID3	ID3 corresponds to common event (`0x23`) STALL_FRONTEND **`0b1`** The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (`0x22`) BR_MIS_PRED_RETIRED **`0b1`** The common event is implemented.	
[1]	ID1	ID1 corresponds to common event (`0x21`) BR_RETIRED **`0b1`** The common event is implemented.	
[0]	ID0	ID0 corresponds to common event (`0x20`) L2D_CACHE_ALLOCATE **`0b1`** The common event is implemented.	

C.3.17 PMCEID2, Performance Monitors Common Event Identification register 2

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x4000 to 0x401F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE28

Reset value

See individual bit resets.

Bit descriptions

Figure C-39: ext_pmceid2 bit assignments

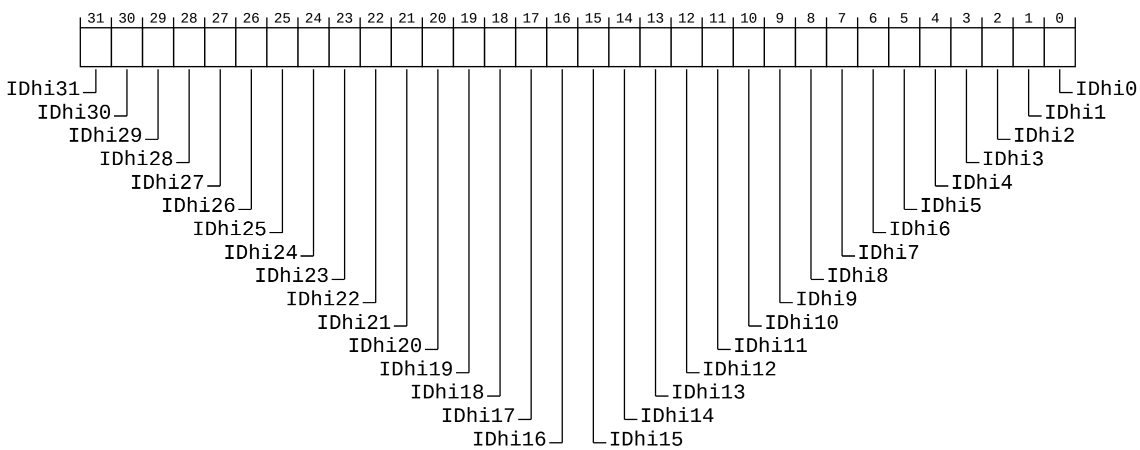


Table C-42: PMCEID2 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x401f) **`0b0`** The common event is not implemented, or not counted.	
[30]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x401e) **`0b0`** The common event is not implemented, or not counted.	
[29]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x401d) **`0b0`** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[28]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x401c) **0b0** The common event is not implemented, or not counted.	
[27]	IDhi27	IDhi27 corresponds to common event (0x401b) CTI_TRIGOUT7 **0b1** The common event is implemented.	
[26]	IDhi26	IDhi26 corresponds to common event (0x401a) CTI_TRIGOUT6 **0b1** The common event is implemented.	
[25]	IDhi25	IDhi25 corresponds to common event (`0x4019`) CTI_TRIGOUT5 **0b1** The common event is implemented.	
[24]	IDhi24	IDhi24 corresponds to common event (`0x4018`) CTI_TRIGOUT4 **0b1** The common event is implemented.	
[23]	IDhi23	IDhi23 corresponds to a Reserved Event event (`0x4017`) **0b0** The common event is not implemented, or not counted.	
[22]	IDhi22	IDhi22 corresponds to a Reserved Event event (`0x4016`) **0b0** The common event is not implemented, or not counted.	
[21]	IDhi21	IDhi21 corresponds to a Reserved Event event (`0x4015`) **0b0** The common event is not implemented, or not counted.	
[20]	IDhi20	IDhi20 corresponds to a Reserved Event event (`0x4014`) **0b0** The common event is not implemented, or not counted.	
[19]	IDhi19	IDhi19 corresponds to common event (`0x4013`) TRCEXTOUT3 **0b1** The common event is implemented.	
[18]	IDhi18	IDhi18 corresponds to common event (`0x4012`) TRCEXTOUT2 **0b1** The common event is implemented.	
[17]	IDhi17	IDhi17 corresponds to common event (`0x4011`) TRCEXTOUT1 **0b1** The common event is implemented.	
[16]	IDhi16	IDhi16 corresponds to common event (`0x4010`) TRCEXTOUT0 **0b1** The common event is implemented.	
[15]	IDhi15	IDhi15 corresponds to common event (0x400f) Reserved **0b0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[14]	IDhi14	IDhi14 corresponds to common event (0x400e) TRB_TRIG **0b0** The common event is not implemented, or not counted.	
[13]	IDhi13	IDhi13 corresponds to common event (0x400d) PMU_OVFS **0b0** The common event is not implemented, or not counted.	
[12]	IDhi12	IDhi12 corresponds to common event (0x400c) TRB_WRAP **0b1** The common event is implemented.	
[11]	IDhi11	IDhi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD **0b1** The common event is implemented.	
[10]	IDhi10	IDhi10 corresponds to common event (0x400a) L2I_CACHE_LMISS **0b0** The common event is not implemented, or not counted.	
[9]	IDhi9	IDhi9 corresponds to common event (`0x4009`) L2D_CACHE_LMISS_RD **0b1** The common event is implemented.	
[8]	IDhi8	IDhi8 corresponds to common event (`0x4008`) Reserved **0b0** The common event is not implemented, or not counted.	
[7]	IDhi7	IDhi7 corresponds to common event (`0x4007`) Reserved **0b0** The common event is not implemented, or not counted.	
[6]	IDhi6	IDhi6 corresponds to common event (`0x4006`) L1I_CACHE_LMISS **0b1** The common event is implemented.	
[5]	IDhi5	IDhi5 corresponds to common event (`0x4005`) STALL_BACKEND_MEM **0b1** The common event is implemented.	
[4]	IDhi4	IDhi4 corresponds to common event (`0x4004`) CNT_CYCLES **0b1** The common event is implemented.	
[3]	IDhi3	IDhi3 corresponds to common event (`0x4003`) SAMPLE_COLLISION **0b1** The common event is implemented.	
[2]	IDhi2	IDhi2 corresponds to common event (`0x4002`) SAMPLE_FILTRATE **0b1** The common event is implemented.	
[1]	IDhi1	IDhi1 corresponds to common event (`0x4001`) SAMPLE_FEED **0b1** The common event is implemented.	

Bits	Name	Description	Reset
[0]	IDhi0	IDhi0 corresponds to common event (`0x4000`) SAMPLE_POP **`0b1`** The common event is implemented.	

C.3.18 PMCEID3, Performance Monitors Common Event Identification register 3

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x4020 to 0x403F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE2C

Reset value

See individual bit resets.

Bit descriptions

Figure C-40: ext_pmceid3 bit assignments

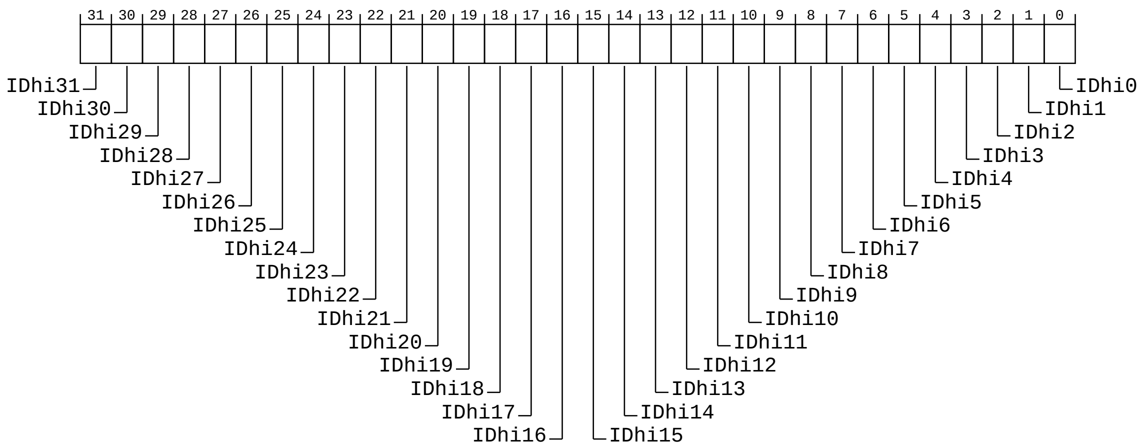


Table C-43: PMCEID3 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x403f) **0b0** The common event is not implemented, or not counted.	
[30]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x403e) **0b0** The common event is not implemented, or not counted.	
[29]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x403d) **0b0** The common event is not implemented, or not counted.	
[28]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x403c) **0b0** The common event is not implemented, or not counted.	
[27]	IDhi27	IDhi27 corresponds to a Reserved Event event (0x403b) **0b0** The common event is not implemented, or not counted.	
[26]	IDhi26	IDhi26 corresponds to a Reserved Event event (0x403a) **0b0** The common event is not implemented, or not counted.	
[25]	IDhi25	IDhi25 corresponds to a Reserved Event event (`0x4039`) **0b0** The common event is not implemented, or not counted.	
[24]	IDhi24	IDhi24 corresponds to a Reserved Event event (`0x4038`) **0b0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[23]	IDhi23	IDhi23 corresponds to a Reserved Event event (`0x4037`) **`0b0`** The common event is not implemented, or not counted.	
[22]	IDhi22	IDhi22 corresponds to a Reserved Event event (`0x4036`) **`0b0`** The common event is not implemented, or not counted.	
[21]	IDhi21	IDhi21 corresponds to a Reserved Event event (`0x4035`) **`0b0`** The common event is not implemented, or not counted.	
[20]	IDhi20	IDhi20 corresponds to a Reserved Event event (`0x4034`) **`0b0`** The common event is not implemented, or not counted.	
[19]	IDhi19	IDhi19 corresponds to a Reserved Event event (`0x4033`) **`0b0`** The common event is not implemented, or not counted.	
[18]	IDhi18	IDhi18 corresponds to a Reserved Event event (`0x4032`) **`0b0`** The common event is not implemented, or not counted.	
[17]	IDhi17	IDhi17 corresponds to a Reserved Event event (`0x4031`) **`0b0`** The common event is not implemented, or not counted.	
[16]	IDhi16	IDhi16 corresponds to a Reserved Event event (`0x4030`) **`0b0`** The common event is not implemented, or not counted.	
[15]	IDhi15	IDhi15 corresponds to a Reserved Event event (0x402f) **`0b0`** The common event is not implemented, or not counted.	
[14]	IDhi14	IDhi14 corresponds to a Reserved Event event (0x402e) **`0b0`** The common event is not implemented, or not counted.	
[13]	IDhi13	IDhi13 corresponds to a Reserved Event event (0x402d) **`0b0`** The common event is not implemented, or not counted.	
[12]	IDhi12	IDhi12 corresponds to a Reserved Event event (0x402c) **`0b0`** The common event is not implemented, or not counted.	
[11]	IDhi11	IDhi11 corresponds to a Reserved Event event (0x402b) **`0b0`** The common event is not implemented, or not counted.	
[10]	IDhi10	IDhi10 corresponds to a Reserved Event event (0x402a) **`0b0`** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[9]	IDhi9	IDhi9 corresponds to a Reserved Event event (`0x4029`) **`0b0`** The common event is not implemented, or not counted.	
[8]	IDhi8	IDhi8 corresponds to a Reserved Event event (`0x4028`) **`0b0`** The common event is not implemented, or not counted.	
[7]	IDhi7	IDhi7 corresponds to a Reserved Event event (`0x4027`) **`0b0`** The common event is not implemented, or not counted.	
[6]	IDhi6	IDhi6 corresponds to common event (`0x4026`) MEM_ACCESS_CHECKED_WR **`0b1`** The common event is implemented.	
[5]	IDhi5	IDhi5 corresponds to common event (`0x4025`) MEM_ACCESS_CHECKED_RD **`0b1`** The common event is implemented.	
[4]	IDhi4	IDhi4 corresponds to common event (`0x4024`) MEM_ACCESS_CHECKED **`0b1`** The common event is implemented.	
[3]	IDhi3	IDhi3 corresponds to common event (`0x4023`) Reserved **`0b0`** The common event is not implemented, or not counted.	
[2]	IDhi2	IDhi2 corresponds to common event (`0x4022`) ST_ALIGN_LAT **`0b1`** The common event is implemented.	
[1]	IDhi1	IDhi1 corresponds to common event (`0x4021`) LD_ALIGN_LAT **`0b1`** The common event is implemented.	
[0]	IDhi0	IDhi0 corresponds to common event (`0x4020`) LDST_ALIGN_LAT **`0b1`** The common event is implemented.	

C.3.19 PMMIR, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE40

Reset value

See individual bit resets.

Bit descriptions

Figure C-41: ext_pmmir bit assignments

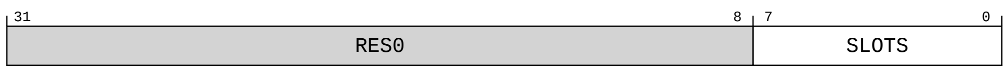


Table C-44: PMMIR bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0x0
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is implemented, this field must not be zero.	

C.3.20 PMDEVARCH, Performance Monitors Device Architecture register

Identifies the programmers' model architecture of the Performance Monitor component.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFBC

Reset value

See individual bit resets.

Bit descriptions

Figure C-42: ext_pmdevarch bit assignments

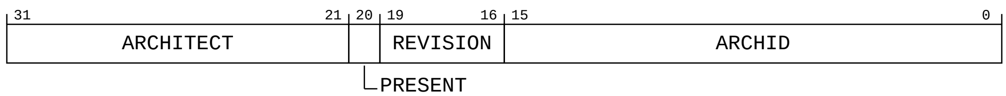


Table C-45: PMDEVARCH bit descriptions

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`.	
[20]	PRESENT	When set to 1, indicates that the DEVARCH is present. This field is 1 in Armv8.	
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. For Performance Monitors, the revision defined by Armv8 is `0x0`. All other values are reserved.	
[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.	

C.3.21 PMDEVID, Performance Monitors Device ID register

Provides information about features of the Performance Monitors implementation.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required from Armv8.2 and in any implementation that includes ARMv8.2-PCSample. Otherwise, its location is RES0.

> **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

Reset value

See individual bit resets.

Bit descriptions

Figure C-43: ext_pmdevid bit assignments

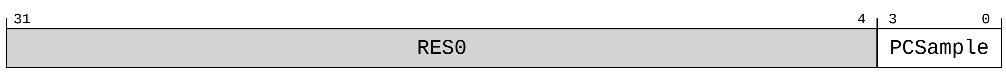


Table C-46: PMDEVID bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	`0x0`
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using Performance Monitors registers. **`0b0001`** PC Sample-based Profiling Extension is implemented in the Performance Monitors register space.	

C.3.22 PMDEVTYPE, Performance Monitors Device Type register

Indicates to a debugger that this component is part of a PEs performance monitor interface.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFCC

Reset value

See individual bit resets.

Bit descriptions

Figure C-44: ext_pmdevtype bit assignments

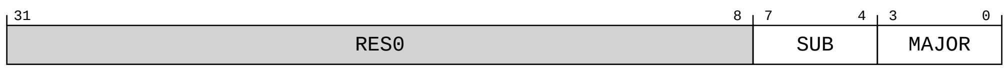


Table C-47: PMDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	SUB	Subtype. Must read as `0x1` to indicate this is a component within a PE.	
[3:0]	MAJOR	Major type. Must read as `0x6` to indicate this is a performance monitor component.	

C.3.23 PMPIDR4, Performance Monitors Peripheral Identification Register 4

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFD0

Reset value

See individual bit resets.

Bit descriptions

Figure C-45: ext_pmpidr4 bit assignments

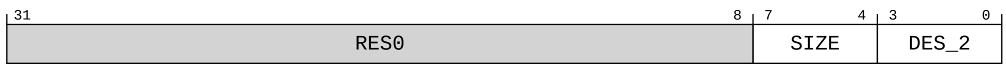


Table C-48: PMPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	SIZE	4KB count. **`0b0000`** The component uses a single 4KB block.	
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0100. **`0b0100`** Arm Limited. This is bits[3:0] of the JEP106 continuation code.	

C.3.24 PMPIDR0, Performance Monitors Peripheral Identification Register 0

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE0

Reset value

See individual bit resets.

Bit descriptions

Figure C-46: ext_pmpidr0 bit assignments

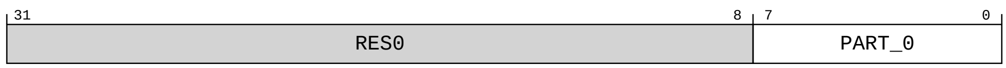


Table C-49: PMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PART_0	Part number, least significant byte. **`0b01001001`** Least significant byte of the PMU unit part.	

C.3.25 PMPIDR1, Performance Monitors Peripheral Identification Register 1

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE4

Reset value

See individual bit resets.

Bit descriptions

Figure C-47: ext_pmpidr1 bit assignments

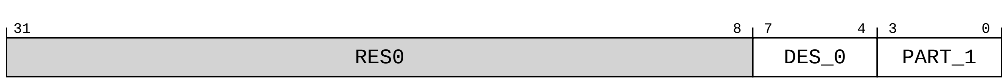


Table C-50: PMPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is 1011. **`0b1011`** Arm Limited. This is the least significant nibble of JEP106 ID code.	
[3:0]	PART_1	Part number, most significant nibble. **`0b1101`** Part number, most significant nibble.	

C.3.26 PMPIDR2, Performance Monitors Peripheral Identification Register 2

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE8

Reset value

See individual bit resets.

Bit descriptions

Figure C-48: ext_pmpidr2 bit assignments

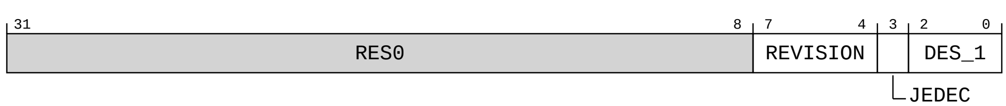


Table C-51: PMPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **`0b0000`** r0p0	
[3]	JEDEC	RAO. Indicates a JEP106 identity code is used. **`0b1`** RES1. Indicates a JEP106 identity code is used	
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 011. **`0b011`** Arm Limited. This is bits[6:4] of the JEP106 ID code.	

C.3.27 PMPIDR3, Performance Monitors Peripheral Identification Register 3

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFEC

Reset value

See individual bit resets.

Bit descriptions

Figure C-49: ext_pmpidr3 bit assignments

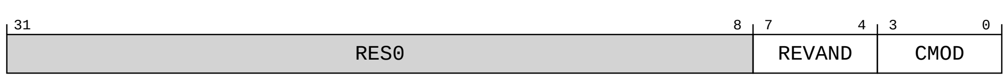


Table C-52: PMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	REVAND	Part minor revision. Parts using ext-PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. `0b0000`	
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. `0b0000` The component is not modified from the original design.	

C.3.28 PMCIDR0, Performance Monitors Component Identification Register 0

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFF0

Reset value

See individual bit resets.

Bit descriptions

Figure C-50: ext_pmcidr0 bit assignments

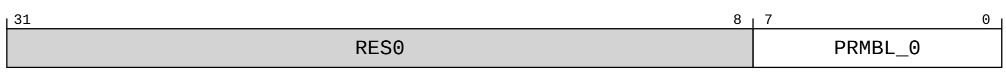


Table C-53: PMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PRMBL_0	Preamble. Must read as `0x0D`. **`0b00001101`** Preamble byte 0	

C.3.29 PMCIDR1, Performance Monitors Component Identification Register 1

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFF4

Reset value

See individual bit resets.

Bit descriptions

Figure C-51: ext_pmcidr1 bit assignments

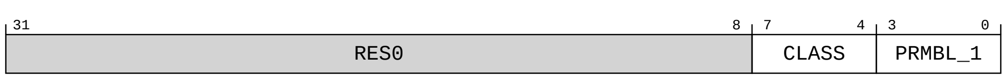


Table C-54: PMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`

Bits	Name	Description	Reset
[7:4]	CLASS	Component class. Reads as `0x9`, debug component. **`0b1001`** Debug Component	
[3:0]	PRMBL_1	Preamble. RAZ. **`0b0000`** Preamble	

C.3.30 PMCIDR2, Performance Monitors Component Identification Register 2

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFF8

Reset value

See individual bit resets.

Bit descriptions

Figure C-52: ext_pmcidr2 bit assignments

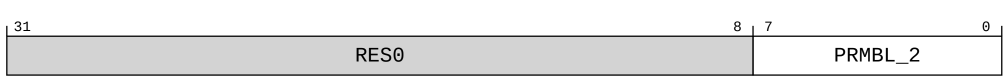


Table C-55: PMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`

Bits	Name	Description	Reset
[7:0]	PRMBL_2	Preamble. Must read as `0x05`. **`0b00000101`** Preamble byte 2.	

C.3.31 PMCIDR3, Performance Monitors Component Identification Register 3

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFFC

Reset value

See individual bit resets.

Bit descriptions

Figure C-53: ext_pmcidr3 bit assignments

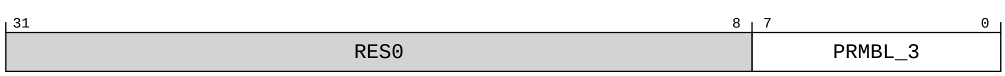


Table C-56: PMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PRMBL_3	Preamble. Must read as `0xB1`. **`0b10110001`** Preamble byte 3.	

C.4 External Debug register summary

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

Table C-57: External Debug register summary

Name	Reset	Width	Description
EDRCR	See individual bit resets.	32-bit	External Debug Reserve Control Register
EDACR	0x0	32-bit	External Debug Auxiliary Control Register
EDPRCR	See individual bit resets.	32-bit	External Debug Power/Reset Control Register
MIDR_EL1	See individual bit resets.	32-bit	Main ID Register
EDPFR	See individual bit resets.	64-bit	External Debug Processor Feature Register
EDDFR	See individual bit resets.	64-bit	External Debug Feature Register
EDDEVARCH	See individual bit resets.	32-bit	External Debug Device Architecture register
EDDEVID2	0x0	32-bit	External Debug Device ID register 2
EDDEVID1	See individual bit resets.	32-bit	External Debug Device ID register 1
EDDEVID	See individual bit resets.	32-bit	External Debug Device ID register 0
EDDEVTYPE	See individual bit resets.	32-bit	External Debug Device Type register
EDPIDR4	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 4
EDPIDR0	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 0
EDPIDR1	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 1
EDPIDR2	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 2
EDPIDR3	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 3
EDCIDR0	See individual bit resets.	32-bit	External Debug Component Identification Register 0
EDCIDR1	See individual bit resets.	32-bit	External Debug Component Identification Register 1
EDCIDR2	See individual bit resets.	32-bit	External Debug Component Identification Register 2
EDCIDR3	See individual bit resets.	32-bit	External Debug Component Identification Register 3

C.4.1 EDRCR, External Debug Reserve Control Register

This register is used to allow imprecise entry to Debug state and clear sticky bits in ext-EDSCR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0x090

Reset value

See individual bit resets.

Bit descriptions

Figure C-54: ext_edrcr bit assignments

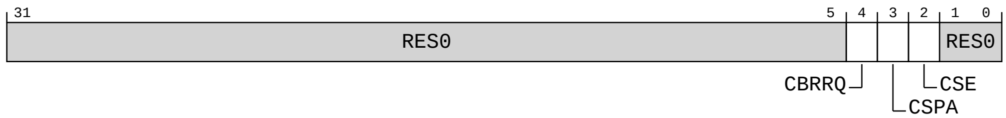


Table C-58: EDRCR bit descriptions

Bits	Name	Description	Reset
[31:5]	RES0	Reserved	`0x0`
[4]	CBRRQ	This feature is not supported. Writes to this bit are ignored **`0b0`** No action. **`0b1`** Allow imprecise entry to Debug state, for example by canceling pending bus accesses.	
[3]	CSPA	Clear Sticky Pipeline Advance. This bit is used to clear the ext-EDSCR.PipeAdv bit to 0. **`0b0`** No action. **`0b1`** Clear the ext-EDSCR.PipeAdv bit to 0.	
[2]	CSE	Clear Sticky Error. Used to clear the ext-EDSCR cumulative error bits to 0. **`0b0`** No action. **`0b1`** Clear the ext-EDSCR.{TXU, RXO, ERR} bits, and, if the PE is in Debug state, the ext-EDSCR.ITO bit, to 0.	
[1:0]	RES0	Reserved	`0b00`

C.4.2 EDACR, External Debug Auxiliary Control Register

Allows implementations to support **IMPLEMENTATION DEFINED** controls.

Configurations

Changing this register from its reset value causes IMPLEMENTATION DEFINED behavior, including possible deviation from the architecturally-defined behavior.

If the EDACR contains any control bits that must be preserved over power down, then these bits must be accessible by the external debug interface when the OS Lock is locked, AArch64-OSLSR_EL1.OSLK == 1, and when the Core is powered off.

Attributes

Width

32

Component

Debug

Register offset

0x094

Reset value

0x0

Bit descriptions

Figure C-55: ext_edacr bit assignments

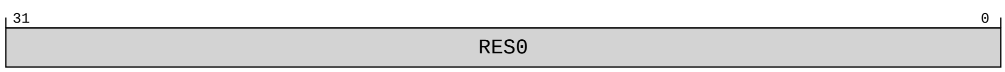


Table C-59: EDACR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0x0

C.4.3 EDPRCR, External Debug Power/Reset Control Register

Controls the PE functionality related to powerup, reset, and powerdown.

Configurations

If ARMv8.3-DoPD is implemented then all fields in this register are in the Core power domain.

CORENPDRQ is the only field that is mapped between the EDPRCR and DBGPRCR and DBGPRCR_EL1.

Attributes

Width

32

Component

Debug

Register offset

0x310

Reset value

See individual bit resets.

Bit descriptions

Figure C-56: ext_edprcr bit assignments

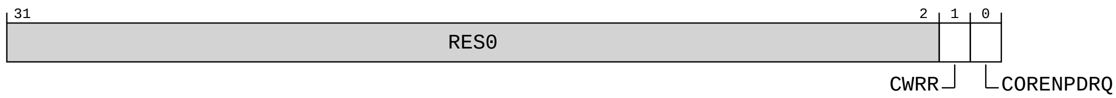


Table C-60: EDPRCR bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	`0x0`
[1]	CWRR	This feature is not supported. Writes to this bit are ignored **`0b0`** No action. **`0b1`** Request Warm reset.	
[0]	CORENPDRQ	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. **`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.	

C.4.4 MIDR_EL1, Main ID Register

Provides identification information for the PE, including an implementer code for the device and a device ID number.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xD00

Reset value

See individual bit resets.

Bit descriptions

Figure C-57: ext_midr_el1 bit assignments

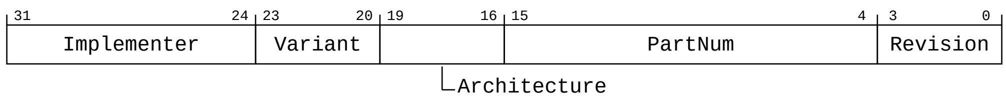


Table C-61: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	Implementer	Indicates the implementer code. This value is: **`0b01000001`** Arm Limited	
[23:20]	Variant	An **IMPLEMENTATION DEFINED** variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product. **`0b0000`** r0p0	
[19:16]	Architecture	Indicates the architecture code. This value is: **`0b1111`** Architecture is defined by ID registers	
[15:4]	PartNum	An **IMPLEMENTATION DEFINED** primary part number for the device. On processors implemented by Arm, if the top four bits of the primary part number are `0x0` or `0x7`, the variant and architecture are encoded differently. **`0b110101001001`** Neoverse N2	
[3:0]	Revision	An **IMPLEMENTATION DEFINED** revision number for the device. **`0b0000`** r0p0	

C.4.5 EDPFR, External Debug Processor Feature Register

Provides information about implemented PE features.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offset

0xD20

Reset value

See individual bit resets.

Bit descriptions

Figure C-58: ext_edpfr bit assignments

63–56	55–52	51–48	47–44	43–40	39–36	35–32
UNKNOWN	RES0	UNKNOWN	AMU	UNKNOWN	SEL2	SVE

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
UNKNOWN	GIC	AdvSIMD	FP	EL3	EL2	EL1	EL0

Table C-62: EDPFR bit descriptions

Bits	Name	Description	Reset
[63:56]	UNKNOWN	Reserved	
[55:52]	RES0	Reserved	0b0000
[51:48]	UNKNOWN	Reserved	
[47:44]	AMU	Activity Monitors Extension. This value is : **0b0001** Activity Monitors Extension version 1 is implemented.	
[43:40]	UNKNOWN	Reserved	
[39:36]	SEL2	Secure EL2. This value is : **0b0001** Secure EL2 is implemented.	
[35:32]	SVE	Scalable Vector Extension. This value is : **0b0001** SVE is implemented.	
[31:28]	UNKNOWN	Reserved	
[27:24]	GIC	System register GIC interface support **0b0011** System register interface to version 4.1 of the GIC CPU interface is supported.	
[23:20]	AdvSIMD	Advanced SIMD. This value is: **0b0001** As for 0b0000, and also includes support for half-precision floating-point arithmetic.	

Bits	Name	Description	Reset
[19:16]	FP	Floating Point. This value is: **0b0001** As for 0b0000, and also includes support for half-precision floating-point arithmetic.	
[15:12]	EL3	AArch64 EL3 Exception level handling **0b0001** EL3 can be executed in AArch64 state only.	
[11:8]	EL2	AArch64 EL2 Exception level handling **0b0001** EL2 can be executed in AArch64 state only.	
[7:4]	EL1	AArch64 EL1 Exception level handling **0b0001** EL1 can be executed in AArch64 state only.	
[3:0]	EL0	AArch64 EL0 Exception level handling **0b0010** EL0 can be executed in both Execution states.	

C.4.6 EDDFR, External Debug Feature Register

Provides top level information about the debug system.

Debuggers must use ext-EDDEVARCH to determine the Debug architecture version.For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offset

0xD28

Reset value

See individual bit resets.

Bit descriptions

Figure C-59: ext_eddfr bit assignments

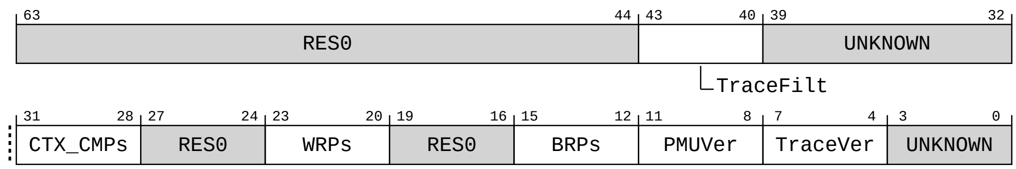


Table C-63: EDDFR bit descriptions

Bits	Name	Description	Reset
[63:44]	RES0	Reserved	`0x0`
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. This value is : **`0b0001`** Armv8.4 Self-hosted Trace Extension is implemented.	
[39:32]	UNKNOWN	Reserved	
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. In an Armv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of AArch64-ID_AA64DFR0_EL1.CTX_CMPs.	
[27:24]	RES0	Reserved	`0b0000`
[23:20]	WRPs	Number of watchpoints, minus 1. The value of 0000 is reserved. In an Armv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of AArch64-ID_AA64DFR0_EL1.WRPs.	
[19:16]	RES0	Reserved	`0b0000`
[15:12]	BRPs	Number of breakpoints, minus 1. The value of 0000 is reserved. In an Armv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of AArch64-ID_AA64DFR0_EL1.BRPs.	
[11:8]	PMUVer	Performance Monitors Extension version.	
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a PE trace unit is implemented. **`0b0001`** PE trace unit System registers implemented.	
[3:0]	UNKNOWN	Reserved	

C.4.7 EDDEVARCH, External Debug Device Architecture register

Identifies the programmers' model architecture of the external debug component.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFBC

Reset value

See individual bit resets.

Bit descriptions

Figure C-60: ext_eddevarch bit assignments

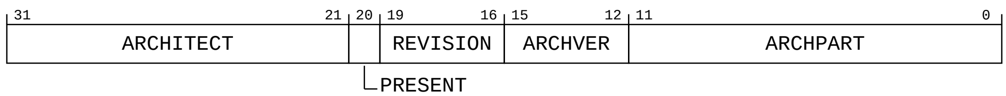


Table C-64: EDDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For debug, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, `0x4`. Bits [27:21] are the JEP106 ID code, `0x3B`.	
[20]	PRESENT	When set to 1, indicates that the DEVARCH is present. This field is 1 in Armv8.	
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. For debug, the revision defined by Armv8-A is `0x0`. All other values are reserved.	
[15:12]	ARCHVER	Defines the architecture version of the component. This is the same value as AArch64-ID_AA64DFR0_EL1.DebugVer and AArch32-DBGDIDR.Version. This value is : **`0b1001`** Armv8.4 Debug architecture.	
[11:0]	ARCHPART	The fields ARCHVER and ARCHPART together form the field ARCHID, so that ARCHPART is ARCHID[11:0]. **`0b101000010101`** The part number of the Armv8-A debug component.	

C.4.8 EDDEVID2, External Debug Device ID register 2

Reserved for future descriptions of features of the debug implementation.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFC0

Reset value

`0x0`

Bit descriptions

Figure C-61: ext_eddevid2 bit assignments

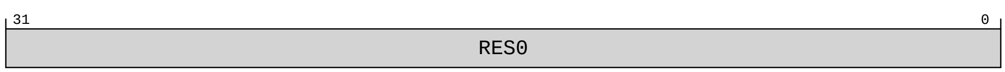


Table C-65: EDDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0x0`

C.4.9 EDDEVID1, External Debug Device ID register 1

Provides extra information for external debuggers about features of the debug implementation.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFC4

Reset value

See individual bit resets.

Bit descriptions

Figure C-62: ext_eddevid1 bit assignments

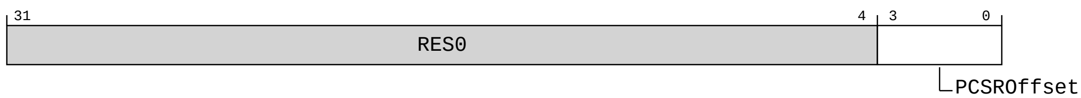


Table C-66: EDDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	`0x0`
[3:0]	PCSROffset	This field indicates the offset applied to PC samples returned by reads of ext-EDPCSR. **`0b0000`** ext-EDPCSR not implemented.	

C.4.10 EDDEVID, External Debug Device ID register 0

Provides extra information for external debuggers about features of the debug implementation.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain.

If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFC8

Reset value

See individual bit resets.

Bit descriptions

Figure C-63: ext_eddevid bit assignments

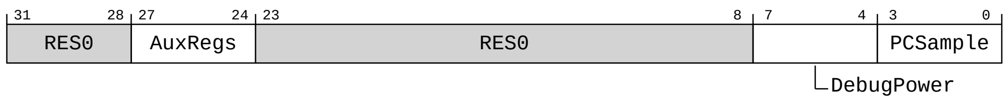


Table C-67: EDDEVID bit descriptions

Bits	Name	Description	Reset
[31:28]	RES0	Reserved	`0b0000`
[27:24]	AuxRegs	Indicates support for Auxiliary registers. **`0b0000`** None supported.	
[23:8]	RES0	Reserved	`0x0`
[7:4]	DebugPower	Indicates support for the ARMv8.3-DoPD feature. **`0b0001`** ARMv8.3-DoPD implemented. All registers in the external debug interface register map are implemented in the Core power domain.	
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using external debug registers. **`0b0000`** PC Sample-based Profiling Extension is not implemented in the external debug registers space.	

C.4.11 EDDEVTYPE, External Debug Device Type register

Indicates to a debugger that this component is part of a PEs debug logic.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFCC

Reset value

See individual bit resets.

Bit descriptions

Figure C-64: ext_eddevtype bit assignments

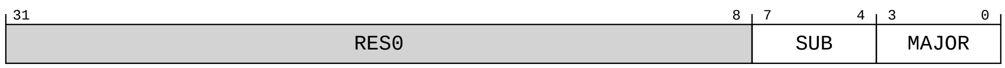


Table C-68: EDDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	SUB	Subtype. Must read as `0x1` to indicate this is a component within a PE.	
[3:0]	MAJOR	Major type. Must read as `0x5` to indicate this is a debug logic component.	

C.4.12 EDPIDR4, External Debug Peripheral Identification Register 4

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFD0

Reset value

See individual bit resets.

Bit descriptions

Figure C-65: ext_edpidr4 bit assignments

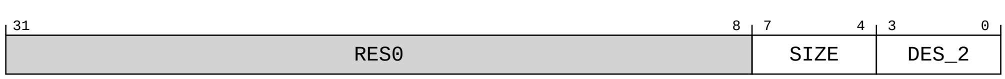


Table C-69: EDPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	SIZE	4KB count. **`0b0000`** The component uses a single 4KB block.	
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0100. **`0b0100`** Arm Limited. This is bits[3:0] of the JEP106 continuation code.	

C.4.13 EDPIDR0, External Debug Peripheral Identification Register 0

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE0

Reset value

See individual bit resets.

Bit descriptions

Figure C-66: ext_edpidr0 bit assignments

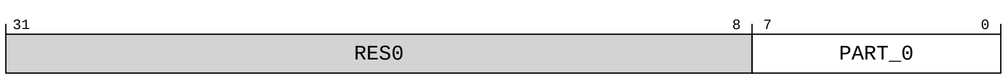


Table C-70: EDPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`

Bits	Name	Description	Reset
[7:0]	PART_0	Part number, least significant byte. **0b01001001** Least Significant byte of the debug part number	

C.4.14 EDPIDR1, External Debug Peripheral Identification Register 1

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE4

Reset value

See individual bit resets.

Bit descriptions

Figure C-67: ext_edpidr1 bit assignments

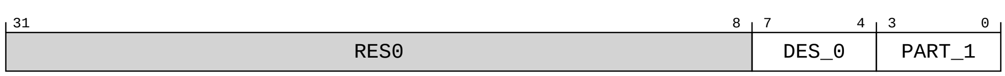


Table C-71: EDPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is 1011. **0b1011** Arm Limited. This is the least significant nibble of JEP106 ID code.	
[3:0]	PART_1	Part number, most significant nibble. **0b1101** Part number, most significant nibble.	

C.4.15 EDPIDR2, External Debug Peripheral Identification Register 2

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE8

Reset value

See individual bit resets.

Bit descriptions

Figure C-68: ext_edpidr2 bit assignments

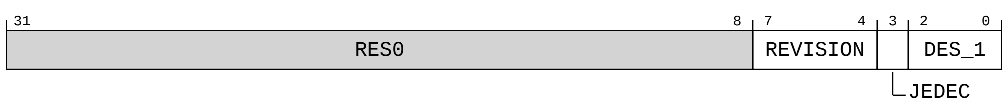


Table C-72: EDPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **`0b0000`** r0p0	
[3]	JEDEC	RAO. Indicates a JEP106 identity code is used. **`0b1`** RAO. Indicates a JEP106 identity code is used	
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 011. **`0b011`** Arm Limited. This is bits[6:4] of the JEP106 ID code.	

C.4.16 EDPIDR3, External Debug Peripheral Identification Register 3

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFEC

Reset value

See individual bit resets.

Bit descriptions

Figure C-69: ext_edpidr3 bit assignments

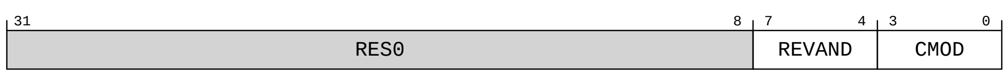


Table C-73: EDPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	REVAND	Part minor revision. Parts using ext-EDPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **`0b0000`**	
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **`0b0000`** The component is not modified from the original design.	

C.4.17 EDCIDR0, External Debug Component Identification Register 0

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFF0

Reset value

See individual bit resets.

Bit descriptions

Figure C-70: ext_edcidr0 bit assignments

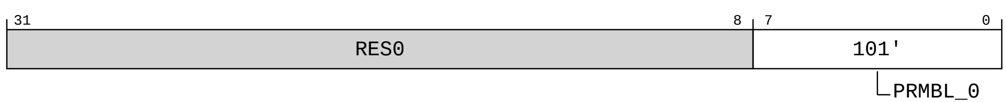


Table C-74: EDCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0x0
[7:0]	PRMBL_0	Preamble.	0b1101

C.4.18 EDCIDR1, External Debug Component Identification Register 1

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFF4

Reset value

See individual bit resets.

Bit descriptions

Figure C-71: ext_edcidr1 bit assignments

Table C-75: EDCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0x0
[7:4]	CLASS	Component class. Debug component.	0b1001
[3:0]	PRMBL_1	Preamble.	0b0

C.4.19 EDCIDR2, External Debug Component Identification Register 2

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFF8

Reset value

See individual bit resets.

Bit descriptions

Figure C-72: ext_edcidr2 bit assignments

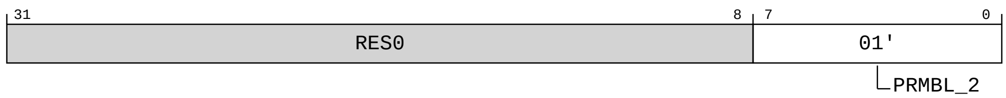


Table C-76: EDCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PRMBL_2	Preamble.	`0b101`

C.4.20 EDCIDR3, External Debug Component Identification Register 3

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

Configurations

If ARMv8.3-DoPD is implemented, this register is in the Core power domain. If ARMv8.3-DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFFC

Reset value

See individual bit resets.

Bit descriptions

Figure C-73: ext_edcidr3 bit assignments

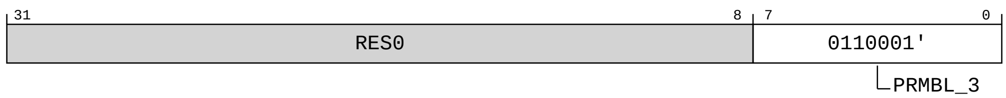


Table C-77: EDCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PRMBL_3	Preamble.	`0b10110001`

C.5 External Activity monitors register summary

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

Table C-78: External Activity monitors register summary

Offset	Name	Reset	Width	Description
0x400	AMEVTYPER00	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x480	AMEVTYPER10	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x484	AMEVTYPER11	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x488	AMEVTYPER12	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x48C	AMEVTYPER13	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0xCE0	AMCGCR	See individual bit resets.	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	See individual bit resets.	32-bit	Activity Monitors Configuration Register
0xE08	AMIIDR	See individual bit resets.	32-bit	Activity Monitors Implementation Identification Register
0xFBC	AMDEVARCH	See individual bit resets.	32-bit	Activity Monitors Device Architecture Register
0xFCC	AMDEVTYPE	See individual bit resets.	32-bit	Activity Monitors Device Type Register
0xFD0	AMPIDR4	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 4
0xFE0	AMPIDR0	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 0
0xFE4	AMPIDR1	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 1
0xFE8	AMPIDR2	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 2
0xFEC	AMPIDR3	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 3
0xFF0	AMCIDR0	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 0
0xFF4	AMCIDR1	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 1
0xFF8	AMCIDR2	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 2
0xFFC	AMCIDR3	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 3

C.5.1 AMEVTYPER00, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR00_EL0 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x400

Reset value

See individual bit resets.

Bit descriptions

Figure C-74: ext_amevtyper00 bit assignments

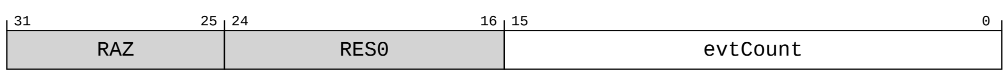


Table C-79: AMEVTYPER00 bit descriptions

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b000000000
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters:	

C.5.2 AMEVTYPER01, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR01_EL0 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x404

Reset value

See individual bit resets.

Bit descriptions

Figure C-75: ext_amevtyper01 bit assignments

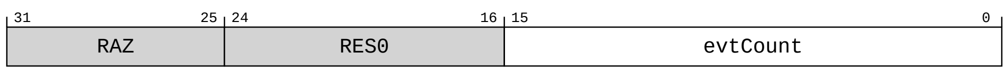


Table C-80: AMEVTYPER01 bit descriptions

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b000000000
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters:	

C.5.3 AMEVTYPER02, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR02_EL0 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x408

Reset value

See individual bit resets.

Bit descriptions

Figure C-76: ext_amevtyper02 bit assignments

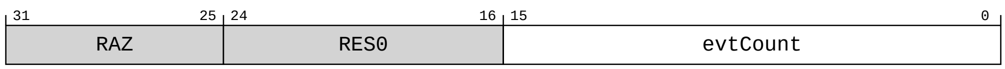


Table C-81: AMEVTYPER02 bit descriptions

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b000000000
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters:	

C.5.4 AMEVTYPER03, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR03_EL0 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x40C

Reset value

See individual bit resets.

Bit descriptions

Figure C-77: ext_amevtyper03 bit assignments

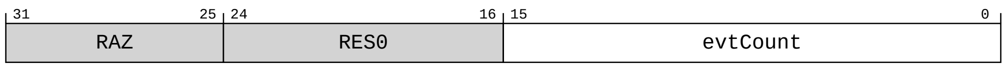


Table C-82: AMEVTYPER03 bit descriptions

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b000000000
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters:	

C.5.5 AMEVTYPER10, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR10_EL0 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x480

Reset value

See individual bit resets.

Bit descriptions

Figure C-78: ext_amevtyper10 bit assignments

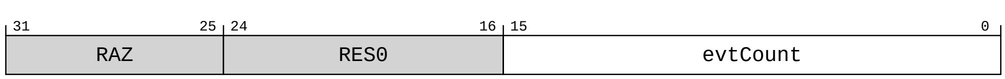


Table C-83: AMEVTYPER10 bit descriptions

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b000000000
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>. It is IMPLEMENTATION DEFINED what values are supported by each counter. If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1<n>, then: • It is UNPREDICTABLE which event will be counted. • The value read back is UNKNOWN. **Note:** The event counted by ext-AMEVCNTR1<n> might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED. If the corresponding counter ext-AMEVCNTR1<n> is enabled, writes to this register have UNPREDICTABLE results.	

C.5.6 AMEVTYPER11, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR11_EL0 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x484

Reset value

See individual bit resets.

Bit descriptions

Figure C-79: ext_amevtyper11 bit assignments

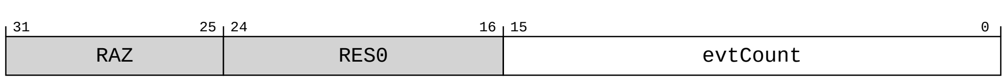


Table C-84: AMEVTYPER11 bit descriptions

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b000000000
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>. It is **IMPLEMENTATION DEFINED** what values are supported by each counter. If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1<n>, then: • It is UNPREDICTABLE which event will be counted. • The value read back is UNKNOWN. **Note:** The event counted by ext-AMEVCNTR1<n> might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED. If the corresponding counter ext-AMEVCNTR1<n> is enabled, writes to this register have UNPREDICTABLE results.	

C.5.7 AMEVTYPER12, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR12_EL0 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x488

Reset value

See individual bit resets.

Bit descriptions

Figure C-80: ext_amevtyper12 bit assignments

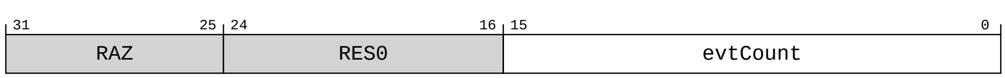


Table C-85: AMEVTYPER12 bit descriptions

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b000000000
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>. It is **IMPLEMENTATION DEFINED** what values are supported by each counter. If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1<n>, then: • It is UNPREDICTABLE which event will be counted. • The value read back is UNKNOWN. **Note:** The event counted by ext-AMEVCNTR1<n> might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED. If the corresponding counter ext-AMEVCNTR1<n> is enabled, writes to this register have UNPREDICTABLE results.	

C.5.8 AMEVTYPER13, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR13_EL0 counts.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0x48C

Reset value

See individual bit resets.

Bit descriptions

Figure C-81: ext_amevtyper13 bit assignments

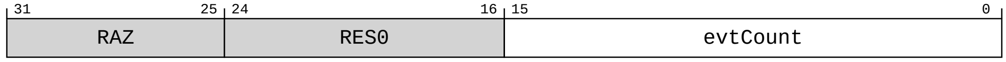


Table C-86: AMEVTYPER13 bit descriptions

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b000000000
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>. It is **IMPLEMENTATION DEFINED** what values are supported by each counter. If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1<n>, then: • It is UNPREDICTABLE which event will be counted. • The value read back is UNKNOWN. **Note:** The event counted by ext-AMEVCNTR1<n> might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED. If the corresponding counter ext-AMEVCNTR1<n> is enabled, writes to this register have UNPREDICTABLE results.	

C.5.9 AMCGCR, Activity Monitors Counter Group Configuration Register

Provides information on the number of activity monitor event counters implemented within each counter group.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xCE0

Reset value

See individual bit resets.

Bit descriptions

Figure C-82: ext_amcgcr bit assignments

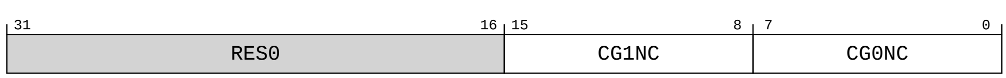


Table C-87: AMCGCR bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	`0x0`
[15:8]	CG1NC	Counter Group 1 Number of Counters. The number of counters in the auxiliary counter group. In AMUv1, the permitted range of values is 0 to 16. **`0b00000011`** Three counters in the auxiliary counter group	
[7:0]	CG0NC	Counter Group 0 Number of Counters. The number of counters in the architected counter group. In AMUv1, the value of this field is 4. **`0b00000100`** Four Counters in the architected counter group	

C.5.10 AMCFGR, Activity Monitors Configuration Register

Global configuration register for the activity monitors.

Provides information on supported features, the number of counter groups implemented, the total number of activity monitor event counters implemented, and the size of the counters. AMCFGR is applicable to both the architected and the auxiliary counter groups.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xE00

Reset value

See individual bit resets.

Bit descriptions

Figure C-83: ext_amcfgr bit assignments

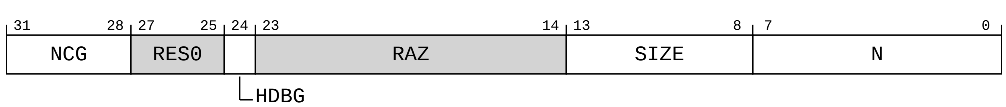


Table C-88: AMCFGR bit descriptions

Bits	Name	Description	Reset
[31:28]	NCG	Defines the number of counter groups. The following value is specified for this product. **0b0001** Two counter groups are implemented	
[27:25]	RES0	Reserved	0b000
[24]	HDBG	Halt-on-debug supported. From Armv8, this feature must be supported, and so this bit is 1. **0b1** ext-AMCR.HDBG is read/write.	
[23:14]	RAZ	Reserved	
[13:8]	SIZE	Defines the size of activity monitor event counters. The size of the activity monitor event counters implemented by the Activity Monitors Extension is defined as [AMCFGR.SIZE + 1]. From Armv8, the counters are 64-bit, and so this field is 111111. **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.	
[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	

C.5.11 AMIIDR, Activity Monitors Implementation Identification Register

Defines the implementer and revisions of the AMU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xE08

Reset value

See individual bit resets.

Bit descriptions

Figure C-84: ext_amiidr bit assignments

31–20	19–16	15–12	11–0
`ProductID`	`Variant`	`Revision`	`Implementer`

Table C-89: AMIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	This field is an AMU part identifier. **`0b110101001001`** Neoverse N2 If ext-AMPIDR0 is implemented, ext-AMPIDR0.PART_0 matches bits [27:20] of this field. If ext-AMPIDR1 is implemented, ext-AMPIDR1.PART_1 matches bits [31:28] of this field.	
[19:16]	Variant	This field distinguishes product variants or major revisions of the product. **`0b0000`** r0p0 If ext-AMPIDR2 is implemented, ext-AMPIDR2.REVISION matches AMIIDR.Variant.	
[15:12]	Revision	This field distinguishes minor revisions of the product. **`0b0000`** r0p0 If ext-AMPIDR3 is implemented, ext-AMPIDR3.REVAND matches AMIIDR.Revision.	
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the AMU. For an Arm implementation, this field reads as `0x43B`. Bits [11:8] contain the JEP106 continuation code of the implementer. Bit 7 is RES0 Bits [6:0] contain the JEP106 identity code of the implementer. If ext-AMPIDR4 is implemented, ext-AMPIDR4.DES_2 matches bits [11:8] of this field. If ext-AMPIDR2 is implemented, ext-AMPIDR2.DES_1 matches bits [6:4] of this field. If ext-AMPIDR1 is implemented, ext-AMPIDR1.DES_0 matches bits [3:0] of this field.	

C.5.12 AMDEVARCH, Activity Monitors Device Architecture Register

Identifies the programmers' model architecture of the AMU component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFBC

Reset value

See individual bit resets.

Bit descriptions

Figure C-85: ext_amdevarch bit assignments

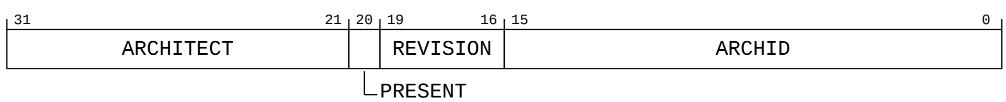


Table C-90: AMDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For AMU, this is Arm Limited.	
[20]	PRESENT	When set to 1, indicates that the DEVARCH is present. **`0b1`** DEVARCH is present	
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. **`0b0000`** Architecture revision is AMUv1. All other values are reserved.	
[15:0]	ARCHID	Defines this part to be an AMU component. For architectures defined by Arm this is further subdivided. For AMU: • Bits [15:12] are the architecture version, `0x0`. • Bits [11:0] are the architecture part number, `0xA66`. This corresponds to AMU architecture version AMUv1.	

C.5.13 AMDEVTYPE, Activity Monitors Device Type Register

Indicates to a debugger that this component is part of a PE's performance monitor interface.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFCC

Reset value

See individual bit resets.

Bit descriptions

Figure C-86: ext_amdevtype bit assignments

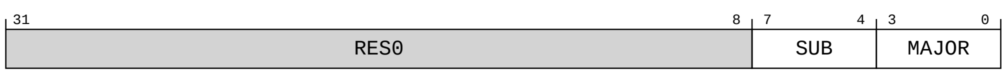


Table C-91: AMDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	SUB	Subtype. Reads as `0x1`, to indicate this is a component within a PE.	
[3:0]	MAJOR	Major type. Reads as `0x6`, to indicate this is a performance monitor component.	

C.5.14 AMPIDR4, Activity Monitors Peripheral Identification Register 4

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFD0

Reset value

See individual bit resets.

Bit descriptions

Figure C-87: ext_ampidr4 bit assignments

31 8	7 4	3 0
RES0	SIZE	DES_2

Table C-92: AMPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0x0
[7:4]	SIZE	4KB count. **0b0000** The component uses a single 4KB block.	
[3:0]	DES_2	Designer. JEP106 continuation code, least significant nibble. The value of this field is IMPLEMENTATION DEFINED. For Arm Limited, this field is 0100. **0b0100** Arm Limited. This is bits[3:0] of the JEP106 continuation code.	

C.5.15 AMPIDR0, Activity Monitors Peripheral Identification Register 0

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE0

Reset value

See individual bit resets.

Bit descriptions

Figure C-88: ext_ampidr0 bit assignments

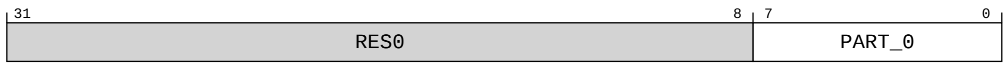


Table C-93: AMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PART_0	Part number, least significant byte. The value of this field is IMPLEMENTATION DEFINED. **`0b01001001`** Part number, least significant byte.	

C.5.16 AMPIDR1, Activity Monitors Peripheral Identification Register 1

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE4

Reset value

See individual bit resets.

Bit descriptions

Figure C-89: ext_ampidr1 bit assignments

31 … 8	7 … 4	3 … 0
RES0	DES_0	PART_1

Table C-94: AMPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. The value of this field is IMPLEMENTATION DEFINED. For Arm Limited, this field is 1011. **`0b1011`** Designer, least significant nibble of JEP106 ID code.	
[3:0]	PART_1	Part number, most significant nibble. The value of this field is IMPLEMENTATION DEFINED. **`0b1101`** Part number, most significant nibble.	

C.5.17 AMPIDR2, Activity Monitors Peripheral Identification Register 2

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE8

Reset value

See individual bit resets.

Bit descriptions

Figure C-90: ext_ampidr2 bit assignments

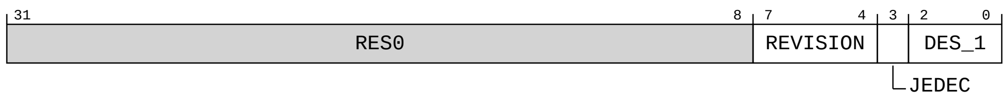


Table C-95: AMPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. The value of this field is IMPLEMENTATION DEFINED. **`0b0000`** r0p0	
[3]	JEDEC	RAO. Indicates a JEP106 identity code is used. **`0b1`** RAO. Indicates a JEP106 identity code is used.	
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. The value of this field is IMPLEMENTATION DEFINED. For Arm Limited, this field is 011. **`0b011`** Arm Limited. This is bits[6:4] of the JEP106 ID code.	

C.5.18 AMPIDR3, Activity Monitors Peripheral Identification Register 3

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFEC

Reset value

See individual bit resets.

Bit descriptions

Figure C-91: ext_ampidr3 bit assignments

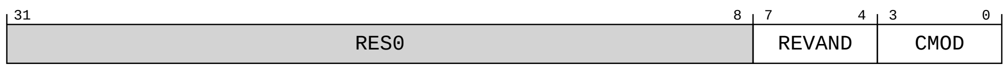


Table C-96: AMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[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. The value of this field is IMPLEMENTATION DEFINED. **`0b0000`**	
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. The value of this field is IMPLEMENTATION DEFINED. **`0b0000`** The component is not modified from the original design.	

C.5.19 AMCIDR0, Activity Monitors Component Identification Register 0

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFF0

Reset value

See individual bit resets.

Bit descriptions

Figure C-92: ext_amcidr0 bit assignments

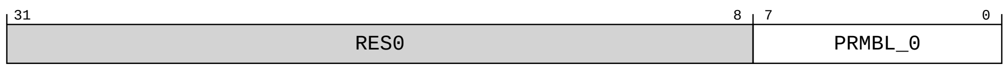


Table C-97: AMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PRMBL_0	Preamble. Must read as `0x0D`. **`0b00001101`** Preamble	

C.5.20 AMCIDR1, Activity Monitors Component Identification Register 1

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFF4

Reset value

See individual bit resets.

Bit descriptions

Figure C-93: ext_amcidr1 bit assignments

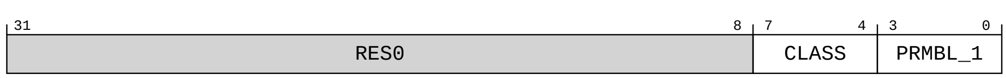


Table C-98: AMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`

Bits	Name	Description	Reset
[7:4]	CLASS	Component class. Reads as `0x9`, CoreSight component. **`0b1001`** CoreSight component.	
[3:0]	PRMBL_1	Preamble. Reads as `0x0`. **`0b0000`** Preamble	

C.5.21 AMCIDR2, Activity Monitors Component Identification Register 2

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFF8

Reset value

See individual bit resets.

Bit descriptions

Figure C-94: ext_amcidr2 bit assignments

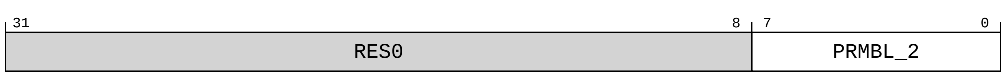


Table C-99: AMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PRMBL_2	Preamble. Reads as `0x05`. **`0b00000101`** Preamble byte 2	

C.5.22 AMCIDR3, Activity Monitors Component Identification Register 3

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFFC

Reset value

See individual bit resets.

Bit descriptions

Figure C-95: ext_amcidr3 bit assignments

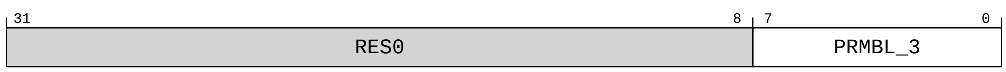


Table C-100: AMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PRMBL_3	Preamble. Reads as `0xB1`. **`0b10110001`** Preamble byte 3	

C.6 External Trace register summary

The summary table provides an overview of all external Trace registers in the core. Individual register descriptions provide detailed information.

Table C-101: External Trace register summary

Offset	Name	Reset	Width	Description
0x018	TRCAUXCTLR	`0x0`	32-bit	Auxillary Control Register
0x180	TRCIDR8	See individual bit resets.	32-bit	ID Register 8

Offset	Name	Reset	Width	Description
0x184	TRCIDR9	See individual bit resets.	32-bit	ID Register 9
0x188	TRCIDR10	See individual bit resets.	32-bit	ID Register 10
0x18C	TRCIDR11	See individual bit resets.	32-bit	ID Register 11
0x190	TRCIDR12	0x0	32-bit	ID Register 12
0x194	TRCIDR13	0x0	32-bit	ID Register 13
0x1C0	TRCIMSPEC0	See individual bit resets.	32-bit	IMP DEF Register 0
0x1E0	TRCIDR0	See individual bit resets.	32-bit	ID Register 0
0x1E4	TRCIDR1	See individual bit resets.	32-bit	ID Register 1
0x1E8	TRCIDR2	See individual bit resets.	32-bit	ID Register 2
0x1EC	TRCIDR3	See individual bit resets.	32-bit	ID Register 3
0x1F0	TRCIDR4	See individual bit resets.	32-bit	ID Register 4
0x1F4	TRCIDR5	See individual bit resets.	32-bit	ID Register 5
0x1F8	TRCIDR6	0x0	32-bit	ID Register 6
0x1FC	TRCIDR7	0x0	32-bit	ID Register 7
0xF00	TRCITCTRL	See individual bit resets.	32-bit	Integration Mode Control Register
0xFA0	TRCCLAIMSET	See individual bit resets.	32-bit	Claim Tag Set Register
0xFA4	TRCCLAIMCLR	See individual bit resets.	32-bit	Claim Tag Clear Register
0xFBC	TRCDEVARCH	See individual bit resets.	32-bit	Device Architecture Register
0xFC0	TRCDEVID2	0x0	32-bit	Device Configuration Register 2
0xFC4	TRCDEVID1	0x0	32-bit	Device Configuration Register 1
0xFC8	TRCDEVID	0x0	32-bit	Device Configuration Register
0xFCC	TRCDEVTYPE	See individual bit resets.	32-bit	Device Type Register
0xFD0	TRCPIDR4	See individual bit resets.	32-bit	Peripheral Identification Register 4
0xFD4	TRCPIDR5	0x0	32-bit	Peripheral Identification Register 5
0xFD8	TRCPIDR6	0x0	32-bit	Peripheral Identification Register 6
0xFDC	TRCPIDR7	0x0	32-bit	Peripheral Identification Register 7
0xFE0	TRCPIDR0	See individual bit resets.	32-bit	Peripheral Identification Register 0
0xFE4	TRCPIDR1	See individual bit resets.	32-bit	Peripheral Identification Register 1
0xFE8	TRCPIDR2	See individual bit resets.	32-bit	Peripheral Identification Register 2
0xFEC	TRCPIDR3	See individual bit resets.	32-bit	Peripheral Identification Register 3
0xFF0	TRCCIDR0	See individual bit resets.	32-bit	Component Identification Register 0
0xFF4	TRCCIDR1	See individual bit resets.	32-bit	Component Identification Register 1
0xFF8	TRCCIDR2	See individual bit resets.	32-bit	Component Identification Register 2
0xFFC	TRCCIDR3	See individual bit resets.	32-bit	Component Identification Register 3

C.6.1 TRCAUXCTLR, Auxillary Control Register

The function of this register is **IMPLEMENTATION DEFINED**.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x018

Reset value

0x0

Bit descriptions

Figure C-96: ext_trcauxctlr bit assignments

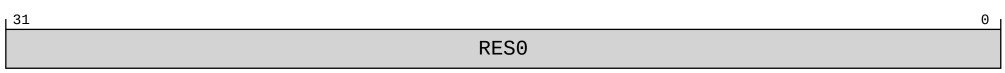


Table C-102: TRCAUXCTLR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0x0

C.6.2 TRCIDR8, ID Register 8

Returns the maximum speculation depth of the instruction trace element stream.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x180

Reset value

See individual bit resets.

Bit descriptions

Figure C-97: ext_trcidr8 bit assignments

Table C-103: TRCIDR8 bit descriptions

Bits	Name	Description	Reset
[31:0]	MAXSPEC	Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of P0 elements in the trace element stream that can be speculative at any time.	

C.6.3 TRCIDR9, ID Register 9

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x184

Reset value

See individual bit resets.

Bit descriptions

Figure C-98: ext_trcidr9 bit assignments

Table C-104: TRCIDR9 bit descriptions

Bits	Name	Description	Reset
[31:0]	NUMP0KEY	Indicates the number of P0 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.	

C.6.4 TRCIDR10, ID Register 10

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x188

Reset value

See individual bit resets.

Bit descriptions

Figure C-99: ext_trcidr10 bit assignments

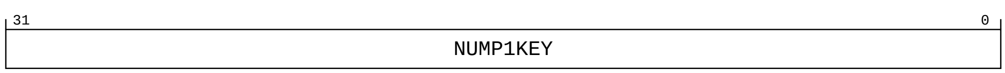


Table C-105: TRCIDR10 bit descriptions

Bits	Name	Description	Reset
[31:0]	NUMP1KEY	Indicates the number of P1 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.	

C.6.5 TRCIDR11, ID Register 11

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x18C

Reset value

See individual bit resets.

Bit descriptions

Figure C-100: ext_trcidr11 bit assignments

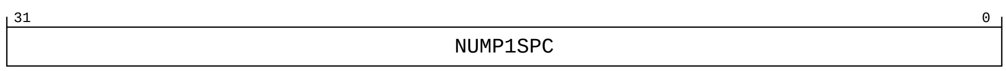


Table C-106: TRCIDR11 bit descriptions

Bits	Name	Description	Reset
[31:0]	NUMP1SPC	Indicates the number of special P1 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.	

C.6.6 TRCIDR12, ID Register 12

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x190

Reset value

`0x0`

Bit descriptions

Figure C-101: ext_trcidr12 bit assignments

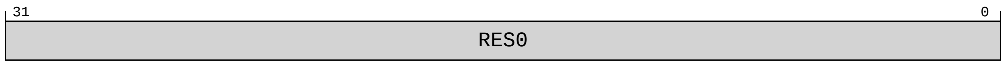


Table C-107: TRCIDR12 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0x0

C.6.7 TRCIDR13, ID Register 13

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x194

Reset value

0x0

Bit descriptions

Figure C-102: ext_trcidr13 bit assignments

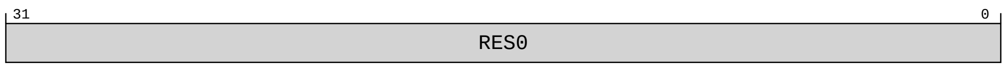


Table C-108: TRCIDR13 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0x0

C.6.8 TRCIMSPEC0, IMP DEF Register 0

TRCIMSPEC0 shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1C0

Reset value

See individual bit resets.

Bit descriptions

Figure C-103: ext_trcimspec0 bit assignments

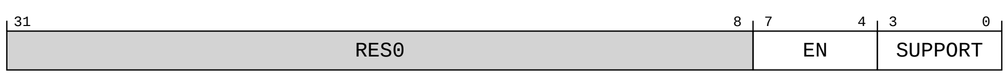


Table C-109: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	EN	Enable. Controls whether the **IMPLEMENTATION DEFINED** features are enabled. **`0b0000`** The IMPLEMENTATION DEFINED features are not enabled. The trace unit must behave as if the IMPLEMENTATION DEFINED features are not supported.	
[3:0]	SUPPORT	Indicates whether the implementation supports **IMPLEMENTATION DEFINED** features. **`0b0000`** No IMPLEMENTATION DEFINED features are supported.	

C.6.9 TRCIDR0, ID Register 0

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1E0

Reset value

See individual bit resets.

Bit descriptions

Figure C-104: ext_trcidr0 bit assignments

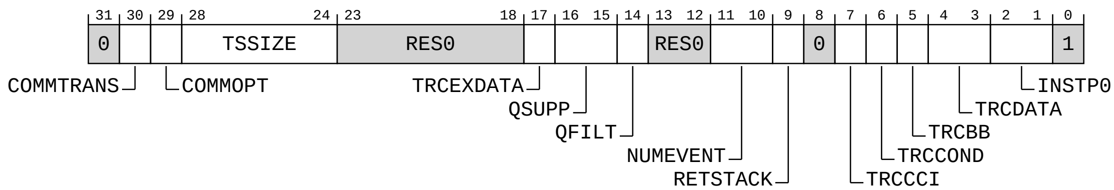


Table C-110: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	0b0
[30]	COMMTRANS	Transaction Start element behavior. **0b0** Transaction Start elements are P0 elements.	
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **0b1** Commit mode 1.	
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. **0b01000** Global timestamping implemented with a 64-bit timestamp value.	
[23:18]	RES0	Reserved	0b000000
[17]	TRCEXDATA	Indicates if the trace unit implements tracing of data transfers for exceptions and exception returns. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures. **0b0** Tracing of data transfers for exceptions and exception returns not implemented. **0b1** Tracing of data transfers for exceptions and exception returns implemented.	

Bits	Name	Description	Reset
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **0b00** Q element support is not implemented.	
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	
[13:12]	RES0	Reserved	0b00
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **0b11** The trace unit supports 4 ETEEvents.	
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **0b1** Return stack implemented.	
[8]	RES0	Reserved	0b0
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **0b1** Cycle counting implemented.	
[6]	TRCCOND	Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b0** Conditional instruction tracing not implemented.	
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. **0b1** Branch broadcasting implemented.	
[4:3]	TRCDATA	Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Data tracing not implemented.	
[2:1]	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.	
[0]	RES1	Reserved	0b1

C.6.10 TRCIDR1, ID Register 1

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1E4

Reset value

See individual bit resets.

Bit descriptions

Figure C-105: ext_trcidr1 bit assignments

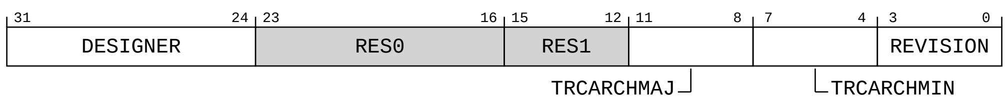


Table C-111: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:24]	DESIGNER	Indicates which company designed the trace unit. The permitted values of this field are the same as AArch64-MIDR_EL1.Implementer. **0b01000001** Arm Limited	
[23:16]	RES0	Reserved	0b00000000
[15:12]	RES1	Reserved	0b1111
[11:8]	TRCARCHMAJ	Major architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	
[7:4]	TRCARCHMIN	Minor architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	
[3:0]	REVISION	Implementation revision that identifies the revision of the trace and OS Lock registers. **0b0010** Revision 2	

C.6.11 TRCIDR2, ID Register 2

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1E8

Reset value

See individual bit resets.

Bit descriptions

Figure C-106: ext_trcidr2 bit assignments

Table C-112: 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.	
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. `0b10` Virtual context identifier selection not supported. ext-TRCCONFIGR.VMIDOPT is RES1.	
[28:25]	CCSIZE	Indicates the size of the cycle counter. `0b0000` The cycle counter is 12 bits in length.	
[24:20]	DVSIZE	Indicates the data value size in bytes. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures. `0b01000` Data value tracing has a maximum of 64-bit data values.	
[19:15]	DASIZE	Indicates the data value size in bytes. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures. `0b01000` Data address tracing has a maximum of 64-bit data addresses.	
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. `0b00100` 32-bit Virtual context identifier size.	
[9:5]	CIDSIZE	Indicates the Context identifier size. `0b00100` 32-bit Context identifier size.	

Bits	Name	Description	Reset
[4:0]	IASIZE	Virtual instruction address size. **0b01000** Maximum of 64-bit instruction address size.	

C.6.12 TRCIDR3, ID Register 3

Returns the base architecture of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1EC

Reset value

See individual bit resets.

Bit descriptions

Figure C-107: ext_trcidr3 bit assignments

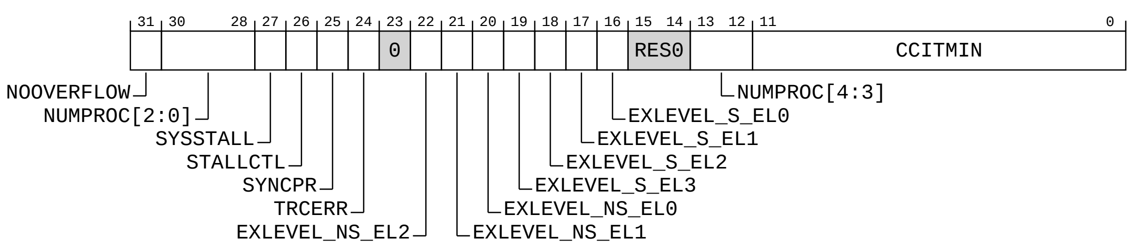


Table C-113: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. **0b00000** The trace unit can trace one PE.	

Bits	Name	Description	Reset
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **0b0** Stalling of the PE is not permitted.	
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **0b0** Stalling of the PE is not implemented.	
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0b0** ext-TRCSYNCPR is read-write so software can change the synchronization period.	
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **0b1** Forced tracing of System Error exceptions is implemented.	
[23]	RES0	Reserved	`0b0`
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 implemented. **0b1** Non-secure EL2 is implemented.	
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 implemented. **0b1** Non-secure EL1 is implemented.	
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 implemented. **0b1** Non-secure EL0 is implemented.	
[19]	EXLEVEL_S_EL3	Indicates if Secure EL3 implemented. **0b1** Secure EL3 is implemented.	
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 implemented. **0b1** Secure EL2 is implemented.	
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 implemented. **0b1** Secure EL1 is implemented.	
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 implemented. **0b1** Secure EL0 is implemented.	
[15:14]	RES0	Reserved	`0b00`
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in ext-TRCCCCTLR.THRESHOLD. If ext-TRCIDR0.TRCCCI == 1 then the minimum value of this field is `0x001`. If ext-TRCIDR0.TRCCCI == 0 then this field is zero.	

C.6.13 TRCIDR4, ID Register 4

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1F0

Reset value

See individual bit resets.

Bit descriptions

Figure C-108: ext_trcidr4 bit assignments

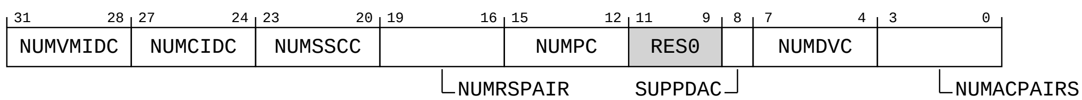


Table C-114: TRCIDR4 bit descriptions

Bits	Name	Description	Reset
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Virtual Context Identifier Comparator.	
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Context Identifier Comparator.	
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. **0b0001** The implementation has one Single-shot Comparator Control.	
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0b0111** The implementation has eight resource selector pairs.	
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0b0000** No PE Comparator Inputs are available.	
[11:9]	RES0	Reserved	0b000

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.	
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. `0b0000` No data value comparators implemented.	
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. `0b0100` The implementation has four Address Comparator pairs.	

C.6.14 TRCIDR5, ID Register 5

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1F4

Reset value

See individual bit resets.

Bit descriptions

Figure C-109: ext_trcidr5 bit assignments

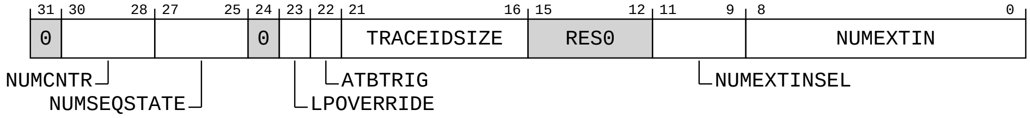


Table C-115: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. **0b010** Two Counters implemented.	
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **0b100** Four Sequencer states are implemented.	
[24]	RES0	Reserved	0b0
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **0b0** The trace unit does not support Low-power Override Mode.	
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **0b1** The implementation supports ATB triggers.	
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **0b000111** The implementation supports a 7-bit trace ID.	
[15:12]	RES0	Reserved	0b0000
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. **0b100** 4 External Input Selector resources are available.	
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **0b111111111** Unified PMU event selection. All other values are reserved.	

C.6.15 TRCIDR6, ID Register 6

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1F8

Reset value

0x0

Bit descriptions

Figure C-110: ext_trcidr6 bit assignments

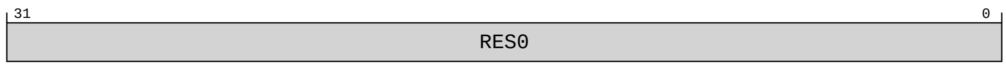


Table C-116: TRCIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0x0

C.6.16 TRCIDR7, ID Register 7

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1FC

Reset value

0x0

Bit descriptions

Figure C-111: ext_trcidr7 bit assignments

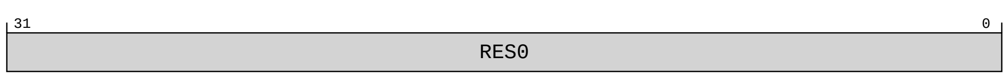


Table C-117: TRCIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0x0

C.6.17 TRCITCTRL, Integration Mode Control Register

A component can use TRCITCTRL to dynamically switch between functional mode and integration mode. In integration mode, topology detection is enabled. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xF00

Reset value

See individual bit resets.

Bit descriptions

Figure C-112: ext_trcitctrl bit assignments

Table C-118: TRCITCTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	`0x0`
[0]	IME	Integration Mode Enable. **`0b0`** The component must enter functional mode. **`0b1`** The component must enter integration mode, and enable support for topology detection and integration testing. This bit is RES0 if no topology detection or integration functionality is implemented.	

C.6.18 TRCCLAIMSET, Claim Tag Set Register

In conjunction with ext-TRCCLAIMCLR, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information see the CoreSight Architecture Specification.

Configurations

The number of claim tag bits implemented is IMPLEMENTATION DEFINED. Arm recommends that implementations support a minimum of four claim tag bits, that is, SET[3:0] reads as 0b1111.

Attributes

Width

32

Component

ETE

Register offset

0xFA0

Reset value

See individual bit resets.

Bit descriptions

Figure C-113: ext_trcclaimset bit assignments

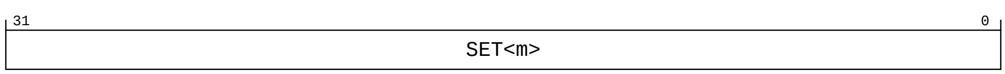


Table C-119: TRCCLAIMSET bit descriptions

Bits	Name	Description	Reset
[31:0]	SET<m>	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 0b1.	

C.6.19 TRCCLAIMCLR, Claim Tag Clear Register

In conjunction with ext-TRCCLAIMSET, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFA4

Reset value

See individual bit resets.

Bit descriptions

Figure C-114: ext_trcclaimclr bit assignments

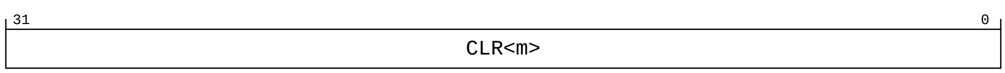


Table C-120: TRCCLAIMCLR bit descriptions

Bits	Name	Description	Reset
[31:0]	CLR<m>	Claim Tag Clear. Indicates the current status of the 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 0b0. The number of Claim Tag bits implemented is indicated in ext-TRCCLAIMSET.	

C.6.20 TRCDEVARCH, Device Architecture Register

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFBC

Reset value

See individual bit resets.

Bit descriptions

Figure C-115: ext_trcdevarch bit assignments

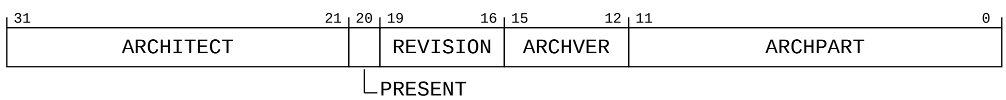


Table C-121: TRCDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code. **`0b01000111011`** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. **`0b1`** Device Architecture information present.	
[19:16]	REVISION	Revision. Defines the architecture revision of the component. **`0b0000`** ETE Version 1.0.	

Bits	Name	Description	Reset
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **`0b0101`** ETE Version 1. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12]. This field reads as `0x5`.	
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **`0b101000010011`** Arm PE trace architecture.	

C.6.21 TRCDEVID2, Device Configuration Register 2

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFC0

Reset value

`0x0`

Bit descriptions

Figure C-116: ext_trcdevid2 bit assignments

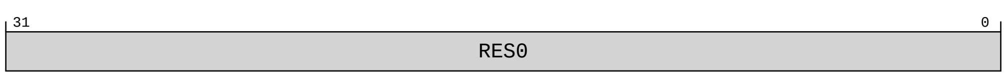


Table C-122: TRCDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0x0`

C.6.22 TRCDEVID1, Device Configuration Register 1

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFC4

Reset value

`0x0`

Bit descriptions

Figure C-117: ext_trcdevid1 bit assignments

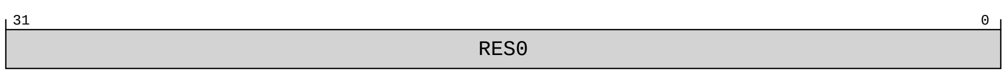


Table C-123: TRCDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0x0`

C.6.23 TRCDEVID, Device Configuration Register

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFC8

Reset value

`0x0`

Bit descriptions

Figure C-118: ext_trcdevid bit assignments

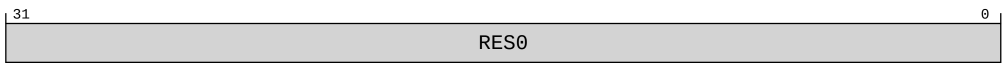


Table C-124: TRCDEVID bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0x0`

C.6.24 TRCDEVTYPE, Device Type Register

Provides discovery information for the component. If the part number field is not recognised, a debugger can report the information that is provided by TRCDEVTYPE about the component instead.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFCC

Reset value

See individual bit resets.

Bit descriptions

Figure C-119: ext_trcdevtype bit assignments

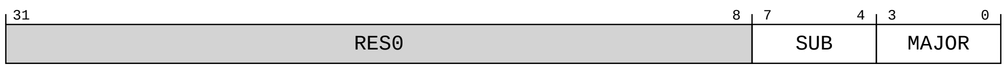


Table C-125: TRCDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	SUB	Component sub-type. **`0b0001`** When MAJOR == 0x3 (Trace source): Associated with a PE. This field reads as `0x1`.	
[3:0]	MAJOR	Component major type. **`0b0011`** Trace source. Other values are defined by the CoreSight Architecture. This field reads as `0x3`.	

C.6.25 TRCPIDR4, Peripheral Identification Register 4

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD0

Reset value

See individual bit resets.

Bit descriptions

Figure C-120: ext_trcpidr4 bit assignments

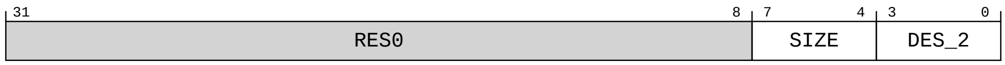


Table C-126: TRCPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	SIZE	The component uses a single 4KB block. **`0b0000`**	
[3:0]	DES_2	Arm Limited. This is bits[3:0] of the JEP106 continuation code. **`0b0100`**	

C.6.26 TRCPIDR5, Peripheral Identification Register 5

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD4

Reset value

`0x0`

Bit descriptions

Figure C-121: ext_trcpidr5 bit assignments

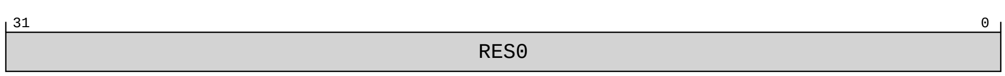


Table C-127: TRCPIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0x0`

C.6.27 TRCPIDR6, Peripheral Identification Register 6

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD8

Reset value

`0x0`

Bit descriptions

Figure C-122: ext_trcpidr6 bit assignments

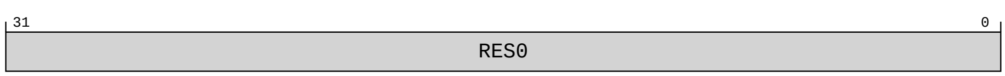


Table C-128: TRCPIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0x0`

C.6.28 TRCPIDR7, Peripheral Identification Register 7

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFDC

Reset value

0x0

Bit descriptions

Figure C-123: ext_trcpidr7 bit assignments

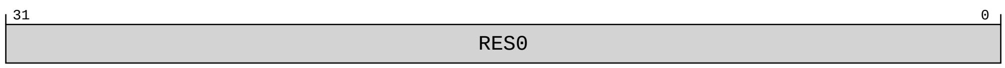


Table C-129: TRCPIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0x0

C.6.29 TRCPIDR0, Peripheral Identification Register 0

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE0

Reset value

See individual bit resets.

Bit descriptions

Figure C-124: ext_trcpidr0 bit assignments

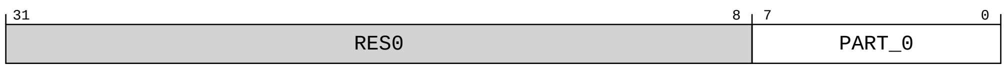


Table C-130: TRCPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PART_0	Least significant byte of the ETM trace unit part. **`0b01001001`**	

C.6.30 TRCPIDR1, Peripheral Identification Register 1

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE4

Reset value

See individual bit resets.

Bit descriptions

Figure C-125: ext_trcpidr1 bit assignments

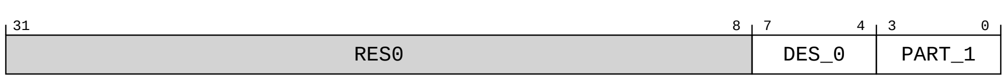


Table C-131: TRCPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`

Bits	Name	Description	Reset
[7:4]	DES_0	Arm Limited. This is the least significant nibble of JEP106 ID code. **0b1011**	
[3:0]	PART_1	Part number, most significant nibble. **0b1101**	

C.6.31 TRCPIDR2, Peripheral Identification Register 2

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE8

Reset value

See individual bit resets.

Bit descriptions

Figure C-126: ext_trcpidr2 bit assignments

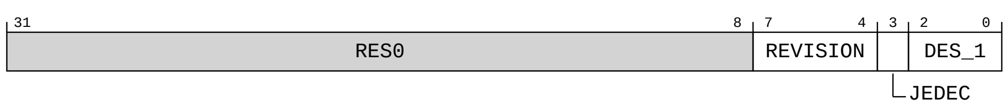


Table C-132: TRCPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	REVISION	r0p0 - Part major revision. **0b0000**	
[3]	JEDEC	JEDEC-assigned JEP106 implementer code is used. **0b1** RES1. Indicates a JEP106 identity code is used	

Bits	Name	Description	Reset
[2:0]	DES_1	Arm Limited. Most significant nibble of JEP106 ID code. **0b011**	

C.6.32 TRCPIDR3, Peripheral Identification Register 3

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFEC

Reset value

See individual bit resets.

Bit descriptions

Figure C-127: ext_trcpidr3 bit assignments

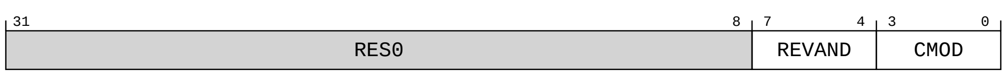


Table C-133: TRCPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0x0
[7:4]	REVAND	Part minor revision. **0b0000**	
[3:0]	CMOD	Not Customer modified. **0b0000**	

C.6.33 TRCCIDR0, Component Identification Register 0

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF0

Reset value

See individual bit resets.

Bit descriptions

Figure C-128: ext_trccidr0 bit assignments

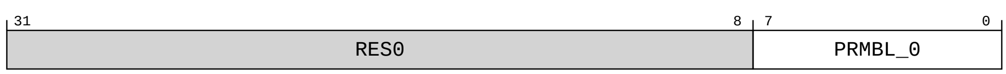


Table C-134: TRCCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PRMBL_0	Component identification preamble, segment 0. **`0b00001101`** Preamble byte 0	

C.6.34 TRCCIDR1, Component Identification Register 1

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF4

Reset value

See individual bit resets.

Bit descriptions

Figure C-129: ext_trccidr1 bit assignments

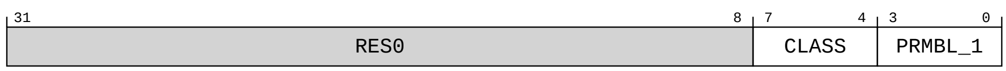


Table C-135: TRCCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:4]	CLASS	Component class. **`0b1001`** CoreSight peripheral.	
[3:0]	PRMBL_1	Component identification preamble, segment 1. **`0b0000`** Preamble	

C.6.35 TRCCIDR2, Component Identification Register 2

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF8

Reset value

See individual bit resets.

Bit descriptions

Figure C-130: ext_trccidr2 bit assignments

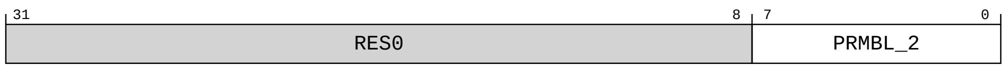


Table C-136: TRCCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PRMBL_2	Component identification preamble, segment 2. **`0b00000101`** Preamble byte 2.	

C.6.36 TRCCIDR3, Component Identification Register 3

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFFC

Reset value

See individual bit resets.

Bit descriptions

Figure C-131: ext_trccidr3 bit assignments

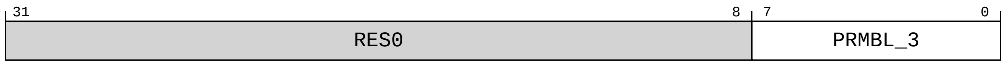


Table C-137: TRCCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0x0`
[7:0]	PRMBL_3	Component identification preamble, segment 3. **`0b10110001`** Preamble byte 3.	

Appendix D Neoverse™ N2 AArch32 UNPREDICTABLE behaviors

There are cases in which the Neoverse™ N2 core implementation diverges from the preferred behavior described in Armv8-A AArch32 **UNPREDICTABLE** behaviors.

D.1 Use of R15 by Instruction

If the use of R15 as a base register for a load or store is **UNPREDICTABLE**, the value used by the load or store using R15 as a base register is the *Program Counter* (PC) with its usual offset and, in the case of T32 instructions, with the forced word alignment. In this case, if the instruction specifies WriteBack, then the load or store is performed without WriteBack.

The Neoverse™ N2 core does not implement a *Read 0* or *Ignore Write* policy on **UNPREDICTABLE** use of R15 by instruction. Instead, the Neoverse™ N2 core takes an **UNDEFINED** exception trap.

D.2 Load/Store accesses crossing page boundaries

The Neoverse™ N2 core implements a set of behaviors for load or store accesses that cross page boundaries.

Crossing a page boundary with different memory types or shareability attributes

The *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*, states that a memory access from a load or store instruction that crosses a page boundary to a memory location that has a different memory type or shareability attribute results in **CONSTRAINED UNPREDICTABLE** behavior.

Crossing a 4KB boundary with a Device access

The *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*, states that a memory access from a load or store instruction to Device memory that crosses a 4KB boundary results in **CONSTRAINED UNPREDICTABLE** behavior.

Implementation (for both page boundary specifications)

For an access that crosses a page boundary, the Neoverse™ N2 core implements the following behaviors:

- Store crossing a page boundary:
    - No alignment fault.
    - The access is split into two stores.
    - Each store uses the memory type and shareability attributes associated with its own address.

- Load crossing a page boundary (Device to Device and Normal to Normal):
  - No alignment fault.
  - The access is split into two loads.
  - Each load uses the memory type and shareability attributes associated with its own address.
- Load crossing a page boundary (Device to Normal and Normal to Device):
  - The instruction generates an alignment fault.

D.3 Armv8-A debug UNPREDICTABLE behaviors

The Neoverse™ N2 core might have Armv8-A debug **UNPREDICTABLE** behaviors either when a topic has multiple options or when the behavior differs from either or both of the Options and Preferences behaviors.

Note

This manual does not describe the behavior when a topic only has a single option and the core implements the preferred behavior.

Table D-1: Armv8 Debug UNPREDICTABLE behaviors

Scenario	Behavior
A32 BKPT instruction with condition code not AL	The core implements the following preferred option: • Executed unconditionally.
Address match breakpoint match only on second halfword of an instruction	The core generates a breakpoint on the instruction if CPSR.IL=0. In the case of CPSR.IL=1, the core does not generate a breakpoint exception.
Address matching breakpoint on A32 instruction with DBGBCRn.BAS=1100	The core implements the following option: • Does match if CPSR.IL=0.
Address match breakpoint match on T32 instruction at DBGBCRn+2 with DBGBCRn.BAS=1111	The core implements the following option: • Does match.
Link to non-existent breakpoint or breakpoint that is not context-aware	The core implements the following option: • No Breakpoint or Watchpoint debug event is generated, and the LBN field of the *linker* reads **UNKNOWN**.
DBGWCRn_EL1.MASK!=00000 and DBGWCRn_EL1.BAS!=11111111	The core behaves as indicated in the sole Preference: • DBGWCRn_EL1.BAS is **IGNORED** and treated as if `0x11111111`.
Address match breakpoint with DBGBCRn_EL1.BAS=0000	The core implements the following option: • As if disabled.
DBGWCRn_EL1.BAS specifies a non-contiguous set of bytes within a double-word	The core implements the following option: • A Watchpoint debug event is generated for each byte.
A32 HLT instruction with condition code not AL	The core implements the following option: • Executed unconditionally.

Scenario	Behavior
Execute instruction at a given EL when the corresponding EDECCR bit is 1 and Halting is allowed	The core behaves as follows: • Generates debug event and Halt no later than the instruction following the next *Context Synchronization operation* (CSO) excluding ISB instruction.
H > N or H = 0 at Non-secure EL1 and EL0, including value read from PMCR_EL0.N	The core implements: • A simple implementation where all of HPMN[4:0] are implemented, and In Non-secure EL1 and EL0: ◦ If H > N then M = N. ◦ If H = 0 then M = 0.
H > N or H = 0: value read back in MDCR_EL2.HPMN	The core implements: • A simple implementation where all of HPMN[4:0] are implemented and for reads of MDCR_EL2.HPMN, return H.
P ≥ M and P ≠ 31: reads and writes of PM XEVTYPER_EL0 and PMXEVCNTR_EL0	The core implements: • A simple implementation where all of SEL[4:0] are implemented, and if P ≥ M and P ≠ 31 then the register is **RES0**.
P ≥ M and P ≠ 31: value read in PMSELR_EL0.SEL	The core implements: • A simple implementation where all of SEL[4:0] are implemented, and if P ≥ M and P ≠ 31 then the register is **RES0**.
P = 31: reads and writes of PMXEVCNTR_EL0	The core implements: • **RES0**.
n ≥ M: Direct access to PMEVCNTRn_EL0 and PMEVTYPERn_EL0	The core implements: • If n ≥ N, then the instruction is **UNALLOCATED**. • Otherwise if n ≥ M, then the register is **RES0**.
Exiting Debug state while instruction issued through EDITR is in flight	The core implements the following option: • The instruction completes in Debug state before executing the restart.
Using memory-access mode with a non-word-aligned address	The core behaves as indicated in the sole Preference: • Does unaligned accesses, faulting if these are not permitted for the memory type.
Access to memory-mapped registers mapped to Normal memory	The core behaves as indicated in the sole Preference: • The access is generated, and accesses might be repeated, gathered, split or resized, in accordance with the rules for Normal memory, meaning the effect is **UNPREDICTABLE**.
Not word-sized accesses or (AArch64 only) doubleword-sized accesses	The core behaves as indicated in the sole Preference: • Reads occur and return **UNKNOWN** data. • Writes set the accessed register(s) to **UNKNOWN**.
External debug write to register that is being reset	The core behaves as indicated in the sole Preference: • Takes reset value.

<table>
<tr><th>Scenario</th><th>Behavior</th></tr>
<tr><td>Accessing reserved debug registers</td><td>The core deviates from preferred behavior because the hardware cost to decode some of these addresses in debug power domain is significantly high.

The actual behavior is:
1. For reserved debug registers in the address range <code>0x000-0xCFC</code> and Performance Monitors registers in the address range <code>0x000</code>, the response is either CONSTRAINED UNPREDICTABLE Error or RES0 when any of the following errors occurs:
Off
The core power domain is either completely off or in a low-power state where the core power domain registers cannot be accessed.
DLK
<code>DoubleLockStatus()</code> is TRUE and OS double-lock is locked (EDPRSR.DLK is 1).
OSLK
OS lock is locked (OSLSR_EL1.OSLK is 1).
2. For reserved debug registers in the address ranges <code>0x400-0x4FC</code> and <code>0x800-0x8FC</code>, the response is CONSTRAINED UNPREDICTABLE Error or RES0 when the conditions in 1 on page 605 do not apply and the following error occurs:
EDAD
<code>AllowExternalDebugAccess()</code> is FALSE. External debug access is disabled.
3. For reserved Performance Monitor registers in the address ranges <code>0x000-0x0FC</code> and <code>0x400-0x47C</code>, the response is either CONSTRAINED UNPREDICTABLE Error, or RES0 when the conditions in 1 on page 605 and 2 on page 605 do not apply, and the following error occurs:
EPMAD
<code>AllowExternalPMUAccess()</code> is FALSE. External Performance Monitors access is disabled.</td></tr>
<tr><td>Clearing the clear-after-read EDPRSR bits when Core power domain is on, and <code>DoubleLockStatus()</code> is TRUE</td><td>The core behaves as indicated in the sole Preference:
• Bits are not cleared to zero.</td></tr>
</table>

D.4 Other UNPREDICTABLE behaviors

The Neoverse™ N2 core implements a set of other **UNPREDICTABLE** behaviors.

Table D-2: Other UNPREDICTABLE behaviors

<table>
<tr><th>Scenario</th><th>Description</th></tr>
<tr><td>CSSELR indicates a cache that is not implemented.</td><td>If CSSELR indicates a cache that is not implemented, then on a read of the CCSIDR the behavior is CONSTRAINED UNPREDICTABLE, and can be one of the following:
• The CCSIDR read is treated as <code>NOP</code>.
• The CCSIDR read is UNDEFINED.
• The CCSIDR read returns an UNKNOWN value (preferred).</td></tr>
</table>

Scenario	Description
HDCR.HPMN is set to 0, or to a value larger than PMCR.N.	If HDCR.HPMN is set to 0, or to a value larger than PMCR.N, then the behavior in Non-secure EL0 and EL1 is **CONSTRAINED UNPREDICTABLE**, and one of the following must happen: • The number of counters accessible is an **UNKNOWN** non-zero value less than PMCR.N. • There is no access to any counters. For reads of HDCR.HPMN by EL2 or higher, if this field is set to 0 or to a value larger than PMCR.N, the core must return a **CONSTRAINED UNPREDICTABLE** value that is one of: • PMCR.N. • The value that was written to HDCR.HPMN. • (The value that was written to HDCR.HPMN) modulo 2h, where h is the smallest number of bits required for a value in the range 0 to PMCR.N.
CRC32 or CRC32C instruction with `size==64`.	On read of the instruction, the behavior is **CONSTRAINED UNPREDICTABLE**, and the instruction executes with the additional decode: `size==32`.
CRC32 or CRC32C instruction with `cond!=1110` in the A1 encoding.	The core implements the following option: • Executed unconditionally.

Appendix E Document revisions

This appendix records the changes between released issues of this document.

E.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 E-1: Issue 0000-02

Change	Location
First Non-Confidential early access release for r0p0	-
Editorial revisions	Throughout document
Added information on Statistical Profiling Extension	22 Statistical Profiling Extension support on page 132 3.1 Core components on page 32 2.1 Neoverse N2 core features on page 23
Updated L2 cache encoding details	10.2 L2 cache encodings on page 77
Added details about PMU events	18.1 Performance monitors events on page 105
Documented registers	16.1 AArch64 random number control register summary on page 97

Table E-2: Differences between Issue 0000-02 and Issue 0000-03

Change	Location
Second Confidential release for r0p0	-
Editorial revisions	Throughout document
Added information on dependent features	2.3 DSU-110 dependent features on page 25
Updated register summary	11.6 AArch64 RAS register summary on page 89
Added TRCIMSPEC0 register details	C.6.8 TRCIMSPEC0, IMP DEF Register 0 on page 572
Added SPE register details	22.4 AArch64 statistical profiling-extension register summary on page 134
PMSSRR register removed	18.4 AArch64 performance monitors register summary on page 116
Added note to clarify power modes	5.4 Core power modes on page 42
Added topic on Write streaming mode	8.5 Write streaming mode on page 62

Table E-3: Differences between Issue 0000-03 and 0000-04

Change	Location
Second Non-Confidential release for r0p0	-
Editorial revisions	Throughout document
Updated DSU product name	Throughout document

Change	Location
Technical updates	16 Random number generator support on page 96
Updated transaction queue sizes	9.3 Transaction capabilities on page 65
Updated supported architecture version	2 The Neoverse N2 core on page 23
Added note to clarify text	17.1.1 Core interfaces on page 99
Updated supported bus ports	2.3 DSU-110 dependent features on page 25
Added note to clarify *Embedded Trace Macrocell* (ETM) export of PMU events	18.1 Performance monitors events on page 105